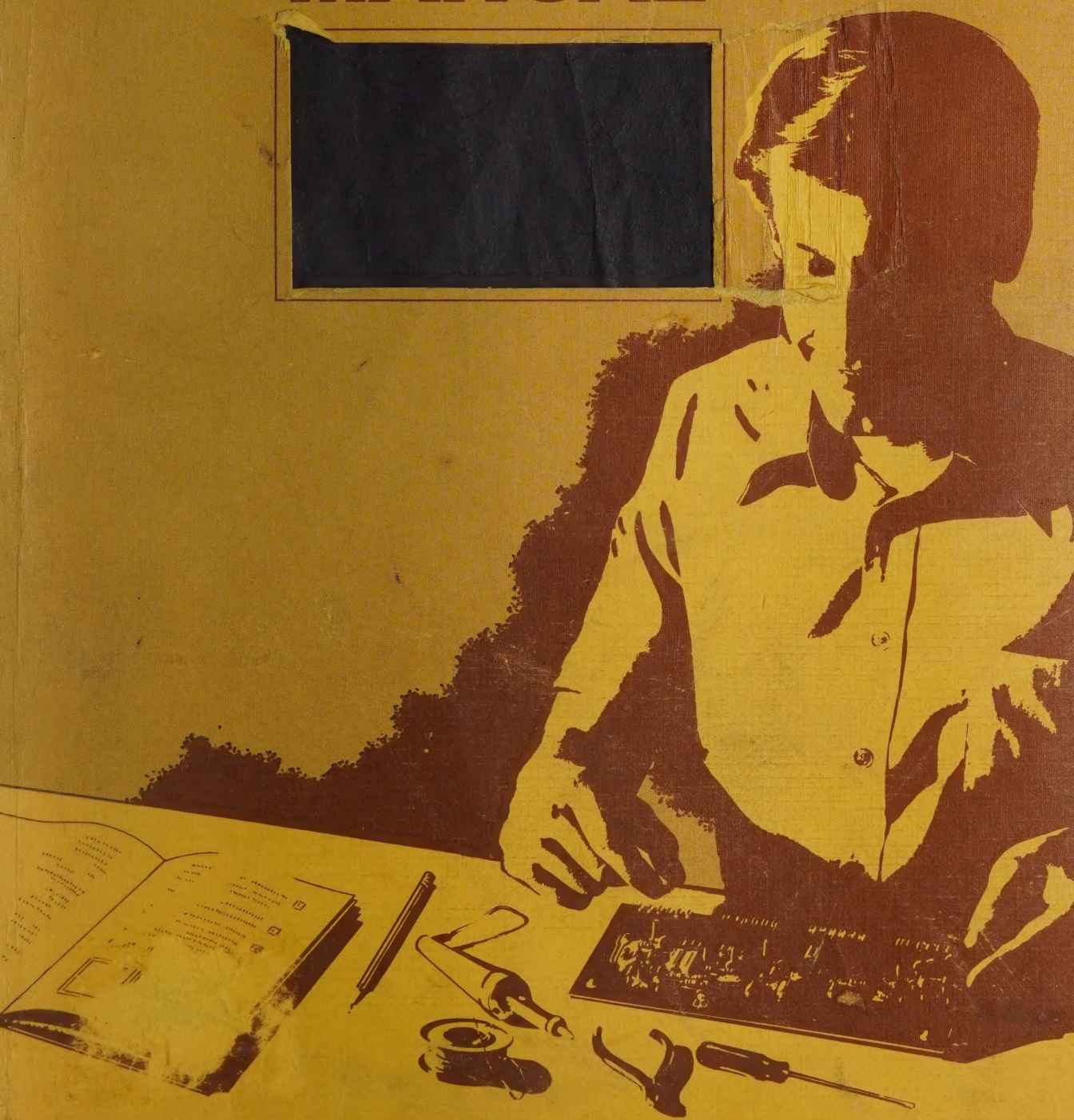
HEATHKIT® MANUAL



HEATH COMPANY • BENTON HARBOR, MICHIGAN

HEATH COMPANY PHONE DIRECTORY

The following telephone numbers are direct lines to the departments listed:

Kit orders and delivery information ..... (616) 982-3411
Credit ..... (616) 982-3561
Replacement Parts ..... (616) 982-3571

Technical Assistance Phone Numbers
8:00 A.M. to 12 P.M. and 1:00 P.M. to 4:30 P.M., EST, Weekdays Only

R/C, Audio, and Electronic Organs ..... (616) 982-3310
Amateur Radio ..... (616) 982-3296
Test Equipment, Weather Instruments and Home Clocks ..... (616) 982-3315
Television ..... (616) 982-3307
Aircraft, Marine, Security, Scanners, Automotive, Appliances and General Products ..... (616) 982-3496
Computers ..... (616) 982-3309

YOUR HEATHKIT 90-DAY LIMITED WARRANTY

For a period of ninety (90) days after purchase, Heath Company will replace or repair free of charge any parts that are defective either in materials or workmanship. You can obtain parts directly from Heath Company by writing us at the address below or by telephoning us at (616) 982-3571. And we'll pay shipping charges to get those parts to you — anywhere in the world.

We warrant that during the first ninety (90) days after purchase, our products, when correctly assembled, calibrated, adjusted and used in accordance with our printed instructions, will meet published specifications.

If a defective part or error in design has caused your Heathkit product to malfunction during the warranty period through no fault of yours, we will service it free upon proof of purchase and delivery at your expense to the Heath factory, any Heathkit Electronic Center, or any of our authorized overseas distributors.

You will receive free consultation on any problem you might encounter in the assembly or use of your Heathkit product. Just drop us a line or give us a call. Sorry, we cannot accept collect calls.

Our warranty does not cover and we are not responsible for damage caused by: incorrect assembly, the use of corrosive solder, defective tools, misuse, or fire; or by unauthorized modifications to or uses of our products for purposes other than as advertised. Our warranty does not include reimbursement for inconvenience, loss of use, customer assembly or set-up time.

This warranty covers only Heathkit products and is not extended to allied equipment or components used in conjunction with our products. **We are not responsible for accidental or consequential damages.** Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

If you are not satisfied with our service (warranty or otherwise) or with our products, write directly to our Director of Customer Services, Heath Company, Benton Harbor, Michigan 49022. He will make certain your problems receive immediate, personal attention.

HEATH COMPANY
BENTON HARBOR, MI. 49022

The Heath Company reserves the right to discontinue products and to change specifications at any time without incurring any obligation to incorporate new features in products previously sold.

Heathkit® Manual

for the

DIGITAL COMPUTER

Model H88

OPERATION 595-2268-02

HEATH COMPANY
BENTON HARBOR, MICHIGAN 49022

Copyright © 1979
Heath Company
All Rights Reserved
Printed in the United States of America

TABLE OF CONTENTS

INTRODUCTION 1-3

SPECIFICATIONS 2-1

SETUP AND TESTING 3-1
Power Line Considerations 3-1
Cabinet Removal 3-2
Testing 3-3

ACCESSORY INSTALLATION 4-1
H88-2 Memory Expansion 4-1
Floppy Disk (H88-1/H17-1) 4-3
H88-3 Serial Interface 4-13
H88-5 Cassette Interface 4-15

SYSTEM CONFIGURATION 5-1
Heath System Configuration 5-1
Terminal Logic Circuit Board 5-1
CPU Logic Circuit Board 5-3
Floppy Disk (H88-1) 5-4
Serial Interface (H88-3) 5-4
Cassette Interface (H88-5) 5-4
Memory Map 5-6
I/O Port Map 5-6
Non-Heath System Configuration 5-7

OPERATION 6-1
Command Summary 6-1
Keyboard Operation 6-2
Normal Modes and Keys 6-4
Special Modes and Keys 6-7
Disk Operation 6-16
Cassette Operation 6-17

READJUSTMENT 7-1

TROUBLESHOOTING 8-1
General Troubleshooting Information 8-2
Troubleshooting Charts 8-3
Shipping Information 8-6

CIRCUIT DESCRIPTION 9-1
Power Supply Circuit Board 9-2
Video Circuit Board 9-3
Terminal Logic Circuit Board 9-6
CPU Logic Circuit Board 9-14
Accessory Circuit Boards 9-18

REPLACEMENT PARTS LIST 10-1
Power Supply 10-1
Video Circuit Board 10-1
Terminal Logic Circuit Board 10-3
CPU Logic Circuit Board 10-3
Chassis Parts 10-4
Floppy Disk Interface Circuit Board 10-5
Serial Interface Circuit Board 10-5
Cassette Interface Circuit Board 10-6

SEMICONDUCTOR IDENTIFICATION 11-1
Component Number Index 11-1
Part Number Index 11-5

APPENDIX 12-1
ASCII Characters 12-1
Graphic Symbols 12-5
Transmitted Codes 12-7
Heath Escape Sequences 12-10
Heath Escape Sequences Defined 12-12
ANSI Escape Sequences 12-17
ANSI Mode Summary 12-19
ANSI Escape Sequences Defined 12-20
The Functions of a Computer 12-27
Instruction Set 12-31
Demonstration Programs 12-50

Z80 CPU 13-1

INS8250 ASYNCHRONOUS
COMMUNICATIONS ELEMENT 14-1

CIRCUIT BOARD X-RAY VIEWS
..... (Illustration Booklet, Page 17)

SCHEMATIC ..... Fold-in

WARRANTY ..... Inside front cover

CUSTOMER SERVICE ..... Inside rear cover

INTRODUCTION

This Heath Digital Computer is a versatile, 8-bit microcomputer and professional video terminal both built into the same cabinet. The computing functions and terminal operations are both controlled by separate Z-80 microprocessors. The high quality keyboard, video display, state-of-the art logic circuitry, and plug-in accessories make this Computer outstanding.

Some features of the Computer are:

- Up to 48k bytes of user-addressable memory.
- Wired and tested CPU and terminal logic circuit boards.
- An internal monitor that automatically sizes the memory and initializes the unit at power-up. The monitor resides in 2k bytes of ROM which contains load and dump routines that eliminate the need for bootstrap and loader programs at turn-on.

The information is displayed on a 12" (diagonal), high-quality cathode-ray tube (CRT) that can display 2000 characters at one time (25 rows of 80 characters). The P4 phosphor used in the CRT provides superb character definition. Upper-case characters are formed by a 5 × 7 dot matrix. Lower-case characters that have descenders use a 5 × 9 dot matrix. The Computer can also display 33 special graphic characters that can be arranged and grouped to form any number of graphic displays and effects. The graphic symbols are formed on an 8 × 10 dot matrix.

Special keyboard and software-controllable escape sequences allow you to select and use many special functions. These include:

- Using either Heath or ANSI escape sequences.
- Eight user-defined special function keys.
- Alternate keypad output (for sending more user-defined special codes to your computer).
- Shifted keypad (so you can obtain the shifted keypad functions without using the SHIFT key).
- Keyboard enable/disable.
- Keyclick enable/disable.
- Cursor-type select (underline or block).
- Auto LF (line feed), auto CR (carriage return).
- Hold screen mode (for scrolling lines and pages).
- Cursor control (left, right, up, down, home).
- Direct cursor addressing.
- Access and use of 25th line.

and you can also:

- Transmit page.
- Transmit 25th line.
- Insert and delete characters and lines.
- Enter and exit the graphics and reverse video modes.
- Erase lines or page of text.
- Modify baud rates.

The highly reliable, standard-size electronic keyboard uses the universally accepted, standard typewriter format. Each key stroke is affirmed by an audible click.

A 12-key keypad duplicates the numeric key in a calculator format. This lets you rapidly enter numbers in programs that call for just numbers. In addition, the shifted keypad functions allow you to insert and delete lines and characters, and move the cursor. Plus, an alternate mode allows you to interchange the shifted and unshifted functions and send special codes to your Computer.

The accessories* allow you to:

- Expand the memory (to 48k of RAM; order model #H88-2 for each 16k).
- Install a floppy disk drive unit (H17-1) and its interface circuit board (H88-1).
- Install a multiport serial interface circuit board (H88-3).
- Install a cassette interface circuit board (H88-5).

These features, along with the stylish molded cabinet, make this Heath Digital Computer a versatile and powerful computing center.

*Some accessories are included with some Computer models.

SPECIFICATIONS

CPU AND MEMORY

Processor	Z-80.
Clock	2.048 MHz.
Memory	16k bytes user RAM (expandable to 48k). 8k System for ROM and RAM. 8k reserved.

DISPLAY

CRT	12″ diagonal, P4 phosphor.
Display Format	24 lines of 80 characters (with access to 25th line).
Display Size	6.5″ high × 8.5″ wide.
Character Size	0.2″ high × 0.1″ wide (approximate).
Character Type	5 × 7 dot matrix (upper case); 5 × 9 dot matrix (lower case with descenders); 8 × 10 dot matrix (graphics).
Keyboard	84 keys (60 alphanumeric, 12 function/control) plus a 12-key numeric/control pad.
Cursor	Blinking, nondestructive underline.
Cursor Controls	Up, down, left, right, home, CR, LF, back space, and tab, and cursor off.
Cursor Addressing	Relative and direct.
Tab	Standard 8-column tab.
Refresh Rate	60 Hz at 60 Hz/50 Hz at 50 Hz line frequency.
Edit Functions	Insert and delete character or line.
Erase Functions	Erase page, erase to end of line, and erase to end of page.
Bell	Audible alarm on receipt of ASCII BEL.
Video	Normal and reverse by character.

ACCESSORIES*

Floppy Disk (H88-4)

Number of Tracks	40.
Sectors Per Track	10.
Formatted Bytes Per Sector	256.
Spindle Motor Speed	300 RPM ±1%.
Data Transfer Rate	128 kHz.
Motor Start Time	1 Sec.

Serial Interface (H88-3)

Channels	2 EIA RS-232C (expandable to 3). Each channel provides serial data and primary RS-232C handshake.
Output Levels	RS-232C.
Input Levels	RS-232C compatible.
Character Length	5, 6, 7, or 8 bits.
Parity	Even, odd, stick, or disabled.
Stop Bits	1, 1-1/2, or 2.
Baud Rates	Continuous, including all standard rates to 19,200 baud.

Cassette Interface (H88-5)

Data Rate	120 bytes/second.
Mark Frequency	2400 Hz.
Space Frequency	1200 Hz.
Audio Output Level (Into High Impedance)	500 mV peak-to-peak or 50 mV peak-to-peak.
Input Sensitivity	500 mV rms.
Speed Tolerance	±33% of recorded speed (wow + flutter + speed difference).
Motor Control	200 VDC maximum.
Contact Ratings	500 mA DC maximum.

GENERAL

Power Requirements	120/240 volts at 90 watts max.
Size	13″ high × 17″ wide × 20″ deep. (33 × 43.2 × 50.8 cm).
Weight	
Operating Temperature	10° to 35° Celcius.
Storage Temperature	0° to 50° Celcius.

The Heath Company reserves the right to discontinue products and to change specifications at any time without incurring any obligation to incorporate new features in products previously sold.

*Some accessories are included with some Computer models.

SET-UP AND TESTING

POWER LINE CONSIDERATIONS

If you need to change the position of the 120/240 switch (located on the bottom of the Computer), be sure you change rear panel fuse F1 to the proper value as follows:

For 120 VAC, use a 1.5-ampere, 125 volt, slow-blow fuse.
For 240 VAC, use a 1-ampere, 250 volt, slow-blow fuse.

The plug on the power cord is for standard 120 VAC outlets. For 240 VAC operation in the U.S.A., cut off and replace in a manner such that your power connection conforms with section 210-21 (b) of the National Electric Code, which reads, in part:

"Receptacles connected to circuits having different voltages, frequencies, or types of current (AC or DC) on the same premises shall be of such design that attachment plugs used on such circuits are not interchangeable."

When you install the new plug, make sure it is connected according to your local electrical code. Units with three-wire line cords must always have the green wire connected to chassis ground.

Be sure the NOR/LOW switch (on the bottom of the Computer) is set in its proper position to match your line voltage as follows:

NOR range — 115 V to 135 V rms or 230 V to 270 V rms.
LOW range — 100 V to 120 V rms or 200 V to 240 V rms.

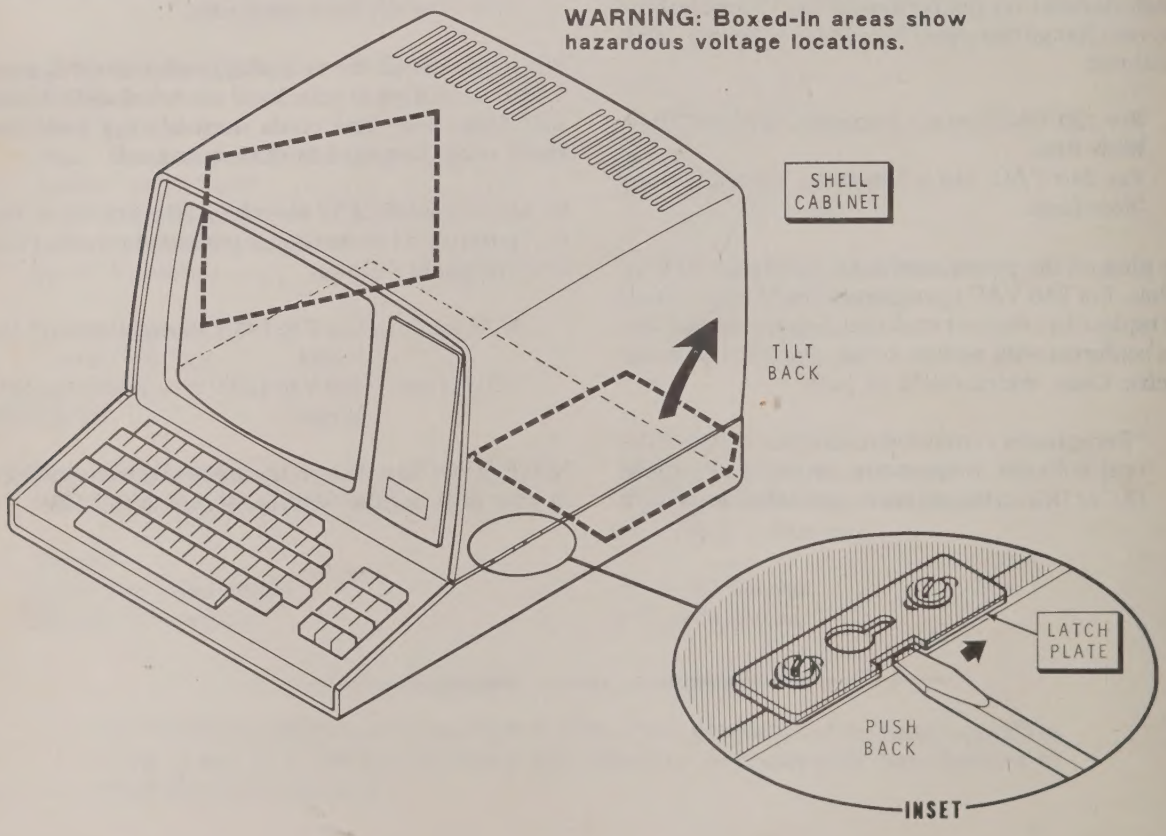
NOTE: If you do not know the value of the line voltage in your area, set the NOR/LOW switch to NOR.

CABINET REMOVAL

Whenever you need to remove the cabinet shell:

- Refer to the inset drawing on Pictorial 3-1, insert the blade of a small screwdriver into the notch in the latch plate, and then slide the latch plate toward the back of the Computer about 1/2".
- Likewise, open the latch plate on the other side of the cabinet shell.
- **WARNING:** When the line cord is connected to an AC outlet, hazardous voltages can be present inside your Computer. See Pictorial 3-1.
- Carefully tilt the cabinet shell back. NOTE: Never allow the cabinet shell to hang unsupported. (The hinges are designed so you can easily remove the cabinet shell from the chassis by sliding the cabinet shell to the left once you have opened it completely.)
- Unplug the fan.

Simply reverse this procedure to close and lock the cabinet shell back on the Computer.



PICTORIAL 3-1

TESTING

The purpose of the "Testing" is to verify that your Computer is working properly. Therefore, it is not necessary, at this time, to have a working knowledge of your Computer. If, at any time during the "Testing," you fail to obtain the proper results, refer to Pictorials 5-1 and 5-2 (Illustration Booklet, Pages 8 and 9) to make sure the circuit board switches are properly set. Then, if necessary, refer to the "Troubleshooting" section on Page 8-1.

INITIAL TESTS

WARNING: Never turn the Computer on and off in rapid succession. This can damage the CRT.

( ) Push the OFF LINE switch once or twice to be sure it is in the out (up) position.

( ) Plug in the line cord and push the rear panel POWER switch to ON. The speaker should produce one or two "beeps."

After a minute or so, the prompt (H:) should appear and the cursor (short line) should blink on and off at the upper left-hand corner of the screen.

( ) Push the OFF LINE key to the down position. Then simultaneously push down the CTRL and G keys. Again, the speaker should "beep."

( ) Press the G key. This time, you should hear only a "tick." (You should hear a tick each time you press a key.) Also, the letter "g" will appear on the screen.

( ) Release the OFF LINE key to its out position.

( ) Push the right-hand SHIFT and RESET keys.

MEMORY TEST

This test will test the user RAM in your Computer. (All other memory is reserved and will not be tested. See the "Memory Map" (Page 5-6) under "Heath System Configuration" on Page 5-1.) The routine will first load a number into each byte of RAM (example: 000 is loaded into each memory location first). Then it will go back and read the number at each location, check it to be sure it is correct, add 001 to the number, place the new number back into the memory location it just read from, and then go on to the next address. This procedure is repeated until all the user RAM is tested for all possibilities, and then starts over.

The test routine first clears the CRT and then displays the memory test message, the last working address (LWA) in the user RAM area (16k = 137377, 32k = 237377, and 48k = 337377), and the pass value (current value stored and read out of RAM).

Example:

```
Dynamic  RAM test
         LWA = 137377 (for 16k of user RAM)
         Pass = 015 (current value stored and read
         out of RAM)
```

If the memory test detects an error before it reaches the end of the memory (LWA), the address where the error occurred and the actual contents that was read will be displayed.

Example:

```
Dynamic  RAM test
         LWA = 137377
         Pass = 015
         ERROR @ 132241 = 017
```

This means that there is a problem with bit 1 (of 0-7) at address 132241. **If the test is successful, the PASS number will reach 377 and then start over.** This will take approximately 5, 10, or 15 minutes; depending on how much memory you have installed.

( ) To start the test, be sure that you have the "H:" prompt. (If not, hold down the right-hand SHIFT key and then push the RESET key. The Computer will reset. Then type G7375↵ (shown as underlined below). [↵ means to push the RETURN (carriage return) key.] The Computer will automatically insert the "0" and spaces, so the entire entry will be:

H: Go 7375↵

To stop the test, simultaneously push the right-hand SHIFT and RESET keys.

Memory failures usually fall into two categories: data and address. A data failure constitutes a particular number or group of numbers from 000 to 377 that cannot be written into or recalled from memory. This type of failure may be due to a solder bridge or defective cells in a memory chip. Since there are eight memory IC's, one for each bit of a byte, it is possible to write a combination of bytes at the address where the test routine failed to determine which, if any, of the memory IC's are at fault. If the memory IC's are interchanged between bits, the difficulty should move with the faulty IC. Be careful when you interchange memory IC's since these IC's are MOS devices. The following chart will help you locate each memory IC.

HIGHEST 16k (H88-2)	U549	U548	U547	U546	U545	U544	U543	U542
MIDDLE 16k (H88-2)	U541	U540	U539	U538	U537	U536	U535	U534
LOWEST 16k	U533	U532	U531	U530	U529	U528	U527	U526
	D7	D6	D5	D4	D3	D2	D1	D0
	MOST SIGNIFICANT DATA DIGIT					LEAST SIGNIFICANT DATA DIGIT		

Address faults are the most difficult to isolate. They may be caused by solder bridges between address lines on the circuit board or by a faulty memory IC. When address lines are shorted together (held high or low), the CPU cannot access the memory locations requested. Often, more than one address will access the location. Therefore, recalling how the "Memory Test" functions, you can see that a given memory location will be incremented too often.

The most practical approach for locating an address failure is to substitute memory IC's or address buffers (U513 and U514), one at a time, until you locate the problem.

COMPUTING TEST

This test will test the computing portion of your Computer. You will first enter a short program into the Computer and then run it. If you make a mistake entering a number, just start over. It is a very short program.

( ) Simultaneously press down the **right-hand** SHIFT key and the RESET key. The Computer will reset and then give you the "H:" prompt.

( ) Type S; the Computer will finish the word "Substitute." Type 040100 and push the RETURN key. The computer will respond with:

040100 XXX__(Each X can by any digit 0-7).

( ) Enter 076 and push the SPACE bar. The new display should be:

040101 XXX __

( ) Enter 011 and push the SPACE bar. The new display should be:

040102 XXX __

( ) Enter 315 and push the SPACE bar.

( ) Continue by making the following entries. Push the SPACE bar only once after each entry.

START	CONTINUE
302	100
003	040
074	303
376	102
155	040
312	

( ) Push the RETURN key. The Computer will respond with the "H:" prompt.

( ) Type P; the Computer will print the words "Program Counter."

Enter 040100 and then push the RETURN key. Again the computer will respond with the "H:" prompt.

In the next step, the Computer will print several symbols, the numbers 0 through 9, several more symbols, the alphabet in capital letters, some more symbols, and part of the alphabet in lower case letters — all on one line. Then it will keep printing this line, and soon will scroll the top line off the page. (NOTE: The bottom line will be flashing.) To stop the program, again push the right-hand SHIFT and RESET keys.

( ) Type G; the computer will print the word "GO." Then push the RETURN key.

NOTES:

1. If your accessories are not installed, proceed to "Accessory Installation" and install them.

2. If your accessories are installed, proceed to the "Configuration" (Page 5-1) and "Operation" (Page 6-1) sections.

ACCESSORY INSTALLATION

NOTE: If you purchased the WH89 Computer (or have no H88 accessories), proceed to "System Configuration" on Page 5-1. Otherwise, perform the following steps that apply to your accessories.

Unplug the line cord and remove the cabinet shell. See Pictorial 3-1 (Page 3-2).

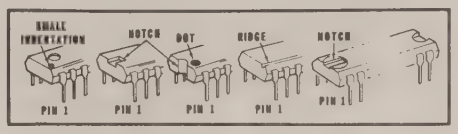
H88-2 MEMORY EXPANSION

The H88-2 Memory Expansion Accessory consists of eight MK4116-4 (#443-904) integrated circuits.

Refer to Pictorial 4-1 (Illustration Booklet, Page 1) for the following steps.

( ) Remove the two screws that hold the CPU logic circuit board in place.

( ) Disconnect the cables from the circuit board and lift the circuit board out of the Computer.

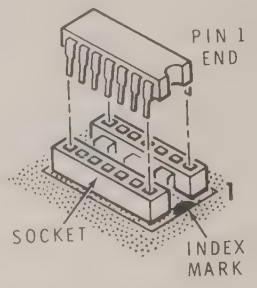


CAUTION: Integrated Circuits (IC's) are complex electronic devices that perform many complicated functions in the circuit. These devices can be damaged during installation. Read all of the following information before you install the IC's.

The pins on the IC's may be bent out at an angle, so they do not line up with the holes in the IC socket. DO NOT try to install an IC without first bending the pins as described below. To do so may damage the IC pins or the socket, causing intermittent contact.

Before you install an IC, lay it down on its side as shown below and very carefully roll it toward the pins to bend the lower pins into line. Then turn the IC over and bend the pins on the other side in the same manner.

Make sure that the pin 1 end of the IC is positioned over the index mark on the circuit board (see the detail at the top of this page). Also make sure that all of the pins are started into the socket. Then press the IC firmly into the socket. NOTE: An IC pin can become bent under the IC and it will appear as though it is correctly installed in the socket.



These memory IC's are called "protected IC's." They can be damaged by static electricity. Once you remove a protected IC from its protective foam packing, DO NOT lay the IC down or let go of it until it is installed in its socket. When you bend the leads of a protected IC, hold the IC in one hand and place your other hand on your work surface before you touch the IC to your work surface. This will equalize the static electricity between the work surface and the IC.

1. ( ) Install eight memory IC's at locations U534 through U541, unless IC's have already been installed there.

2. ( ) If you still have eight memory IC's not installed, install them at locations U542 through U549.

— If you performed step 2 above (for a total of 48k of memory), set the indicated jumpers as follows:

JUMPER	TO
JJ504	0
JJ503	0
JJ502	1
JJ501	0

— If you performed only step 1 above (for a total of 32k of memory), set the indicated jumpers as follows:

JUMPER	TO
JJ504	0
JJ503	0
JJ502	0
JJ501	1

— If for some reason you remove memory (place the IC's in protective foam) and have only the basic 16k installed (IC's U526 through U533), set the indicated jumpers as follows:

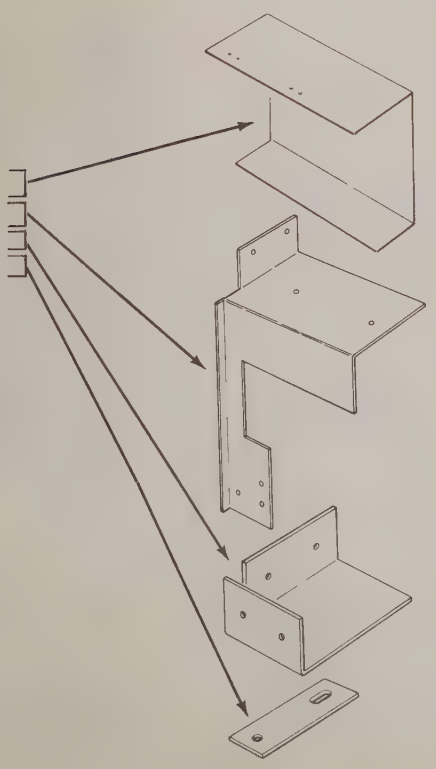
JUMPER	TO
JJ504	0
JJ503	0
JJ502	0
JJ501	0

( ) Now reinstall and reconnect the CPU logic circuit board, reinstall the cabinet shell, (unless you intend to install other accessories now), and reperform the "Memory Test" on Page 3-3.

FLOPPY DISK (H88-4)

PARTS LIST

HEATH Part No.	QTY.	DESCRIPTION
181-3055	1	Wired floppy disk interface circuit board
134-1074	1	Ribbon cable
206-1331	1	Mu-Metal shield
204-2420	1	Top floppy bracket
204-2421	1	Bottom floppy bracket
205-1804	1	Support bracket
442-30	1	uA309k 5-volt regulator
442-650	1	78H12 12-volt regulator
443-764	2	2114 IC
444-19	1	2316 IC
75-124	1	Large paper insulator
75-108	3	Small paper insulator
346-64	4"	Large sleeving
250-56	1	6-32 × 1/4" screw
250-1264	3	6-32 × 3/8" hex head screw
254-1	1	#6 lockwasher
250-137	1	8-32 × 3/8" screw
254-2	1	#8 lockwasher
255-47	1	3/8" spacer
390-1675	1	H89 label
352-31	1	Thermal compound
448-258	1	Diskette
	1	H17-1 floppy drive unit



CPU LOGIC CIRCUIT BOARD

( ) Turn the Computer off, unplug the line cord and remove the cabinet shell. See Pictorial 3-1 on Page 3-2.

Refer to Pictorial 4-2 (Illustration Booklet, Page 2) for the following steps.

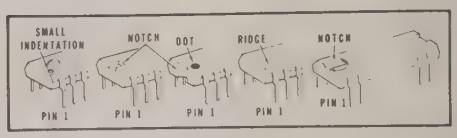
( ) Remove the two screws that hold the CPU logic circuit board in place.

( ) Disconnect the cables from the circuit board and lift the circuit board up out of the Computer.

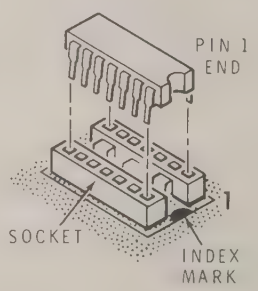
CAUTION: Integrated Circuits (IC's) are complex electronic devices that perform many complicated functions in the circuit. These devices can be damaged during installation. Read all of the following information before you install the IC's.

The pins on the IC's may be bent out at an angle, so they do not line up with the holes in the IC socket. DO NOT try to install an IC without first bending the pins as described below. To do so may damage the IC pins or the socket, causing intermittent contact.

Before you install an IC, lay it down on its side as shown below and very carefully roll it toward the pins to bend the lower pins into line. Then turn the IC over and bend the pins on the other side in the same manner.



Make sure that the pin 1 end of the IC is positioned over the index mark on the circuit board (see the detail at the top of this page). Also make sure that all of the pins are started into the socket. Then press the IC firmly into the socket. NOTE: An IC pin can become bent under the IC and it will appear as though it is correctly installed in the socket.



These IC's are called "protected IC's." They can be damaged by static electricity. Once you remove a protected IC from its protective foam packing, DO NOT lay the IC down or let go of it until it is installed in its socket. When you bend the leads of a protected IC, hold the IC in one hand and place your other hand on your work surface before you touch the IC to your work surface. This will equalize the static electricity between the work surface and the IC.

Install the three IC's as follows:

( ) U523: 2114 (#443-764).

( ) U525: 2114 (#443-764).

( ) U520: 2316 (#444-19).

( ) Check to be sure that Jumper JJ508 is at position B as shown (connected to lower two pins).

( ) Cut a 3/4" × 3/4" piece of insulating paper from one of the small insulating papers supplied.

( ) Remove the protective paper backing from the insulating paper and press the insulating paper against the side of crystal X502 as shown.

Refer to Pictorial **4-3** (Illustration Booklet, Page 3) for the following steps.

( ) Remove the two screws that hold the right-hand accessory mounting bracket and set the bracket aside temporarily.

( ) Refer to the inset drawing, twist the small-plug end of the floppy cable as shown, and slide the length of large sleeving over the cable and position it as shown in the Pictorial.

( ) Be sure the marked edge of the floppy cable is positioned as shown. Then push the smaller plug into the connector of the floppy disk I/O circuit board. NOTE: The edge may be marked by a special color stripe or a small triangle on the connector.

( ) Plug the floppy circuit board into the CPU circuit board at P506 and P512. Be sure the insulating paper on crystal X502 is between the crystal and the cutoff pins on the floppy disk I/O circuit board.

( ) Remount the accessory mounting bracket. Be sure the nylon guide holds the edge of the circuit board. Then set this assembly aside temporarily.

POWER SUPPLY

Refer to Pictorial **4-4** (Illustration Booklet, Page 3) for the following steps.

( ) Remove the two screws that hold socket U103 and set the screws aside.

WARNING: You will be using Dow Corning 340 thermal compound in the following steps. Although this compound is not caustic, it may cause temporary discomfort if it gets into your eyes. If this happens, rinse your eyes with warm water. If the compound gets on your clothing, it may require professional cleaning, so handle the compound carefully. This compound contains zinc oxides, SiO_2 and slight traces of CO_2.

( ) Locate the thermal compound pod and make a small slit in one side.

( ) Spread a small amount of thermal compound onto the 78H12 integrated circuit (#442-650). Squeeze the pod to remove the compound.

( ) U103: Notice the wide space of the IC and install the IC in socket U103. Be sure the socket is properly seated in its mounting holes in the heat sink and then install the two screws you removed earlier.

( ) U102: In the same manner, install the μA309K IC (#442-30) at U102.

If you have a voltmeter, make the following tests. Otherwise, proceed to "CPU Board Reinstallation."

( ) Connect the common voltmeter lead to the heat sink as shown.

NOTE: If you do not obtain the proper results in the following steps, immediately unplug the line cord and proceed to "Troubleshooting" on Page 8-1. Be sure to correct the problem before you proceed.

( ) Plug in the line cord and turn on the Computer.

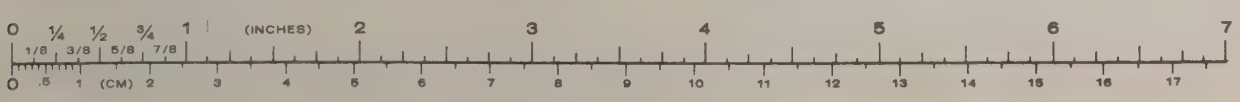
( ) Measure TP1. The voltage should be + 4.5 to + 5.5 volts DC.

( ) Measure TP2. The voltage should be + 10.8 to + 13.2 volts DC.

( ) Turn off the Computer and unplug the line cord. Then disconnect the voltmeter and set it aside.

CPU Board Reinstallation

( ) Remount and reconnect the CPU logic circuit board. See Pictorial 4-2 (Illustration Booklet, Page 2).



Refer to Pictorial 4-5 (Illustration Booklet, Page 4) for the following steps.

( ) Unpack your floppy drive unit (H17-1).

The Floppy Disk drive unit supplied to you may have one of two different circuit boards. Parts A and B of Pictorial 4-5 (Illustration Booklet, Page 4) show portions of these two circuit boards. Examine the circuit board on the Floppy Disk drive unit that you have. If the programming plug has eight jumper strips, refer to Part A of Pictorial 4-5 for the following steps. If the programming plug has only six jumper strips, refer to Part B of Pictorial 4-5 for the following steps. It is a good idea to circle or put a check mark beside the appropriate part (A or B) of Pictorial 4-5 that pertains to your particular drive unit.

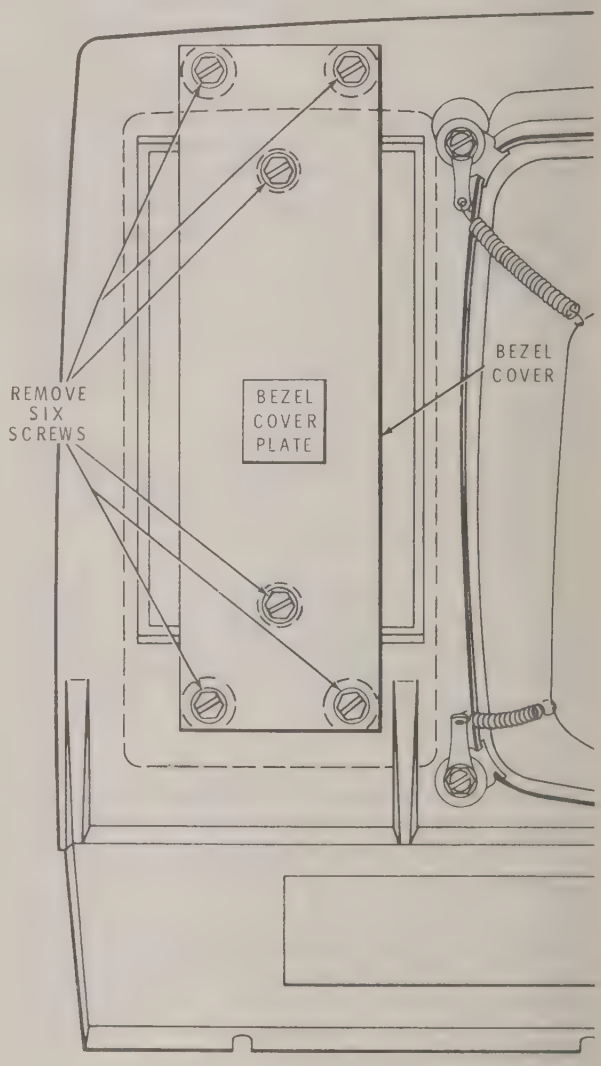
( ) Use a knife or other suitable tool and cut strips DS1 and DS2 of the programming plug.

( ) Examine the circuit board foil labeled "MUX" (shown in Part A of Pictorial 4-5) or the "MX" jumper strip (shown in Part B). If your drive unit has an uncut foil or jumper strip, use a knife or other suitable tool and very carefully cut a small gap in the foil or jumper strip.

NOTE: This drive unit is referred to in software as SYØ:.

Refer to Pictorial 4-6 (Illustration Booklet, Page 4) for the following steps.

( ) Refer to Detail 4-6A and remove the six screws that hold the bezel cover and bezel cover plate. Set the bezel and plate aside.You will use the screws in the following steps.

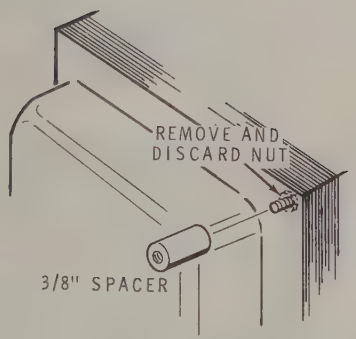


Detail 4-6A

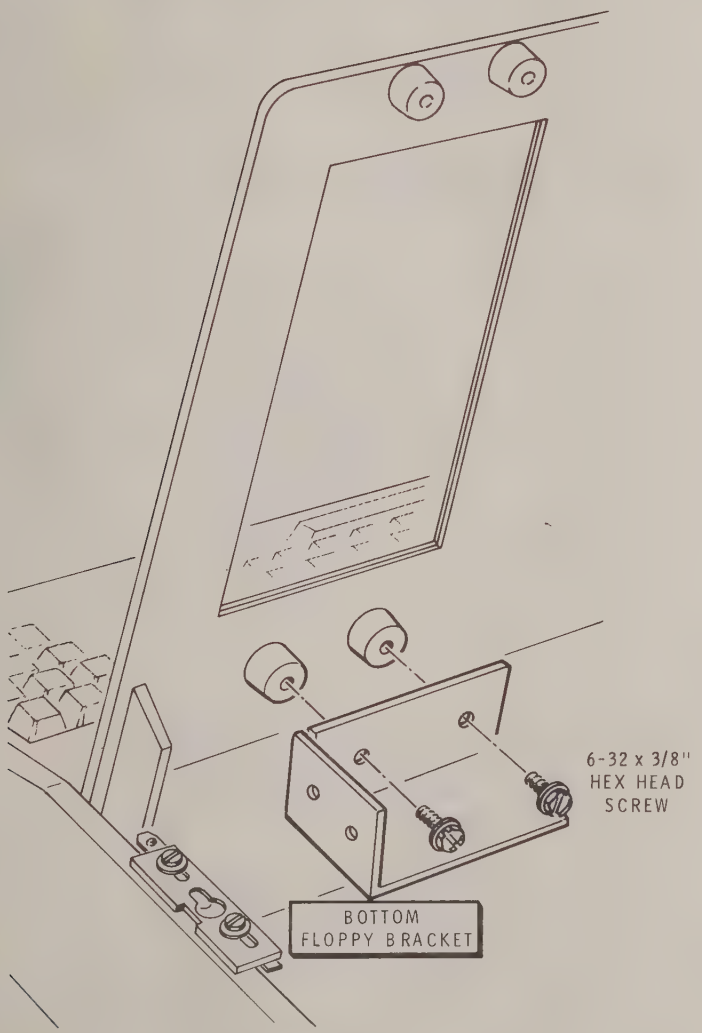
( ) Refer to the inset drawing on Pictorial 4-6, remove the protective backing from the H89 label, and press the label in place on the front of the Computer.

( ) Refer to Detail 4-6B and remove and discard the nut from the transformer at T1. Then install a 3/8″ spacer at T1.

( ) Refer to Detail 4-6C and mount the bottom floppy bracket to the front panel with two 6-32 × 3/8″ hex head screws.



Detail 4-6B



Detail 4-6C

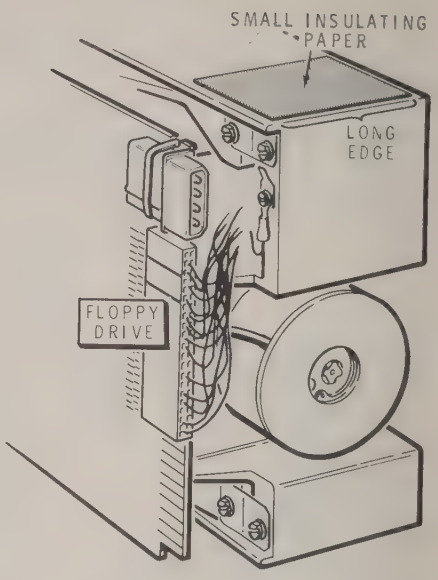
( ) Refer to Detail 4-6D and mount the top floppy bracket to the front panel with two 6-32 × 3/8″ hex head screws and to the bottom floppy bracket with two 6-32 × 3/8″ hex head screws. Use screws you removed earlier as necessary but **DO NOT** tighten the screws at this time.

( ) Refer to Detail 4-6E, remove the protective backing from a piece of small insulating paper, and press the paper onto the floppy drive unit as shown.

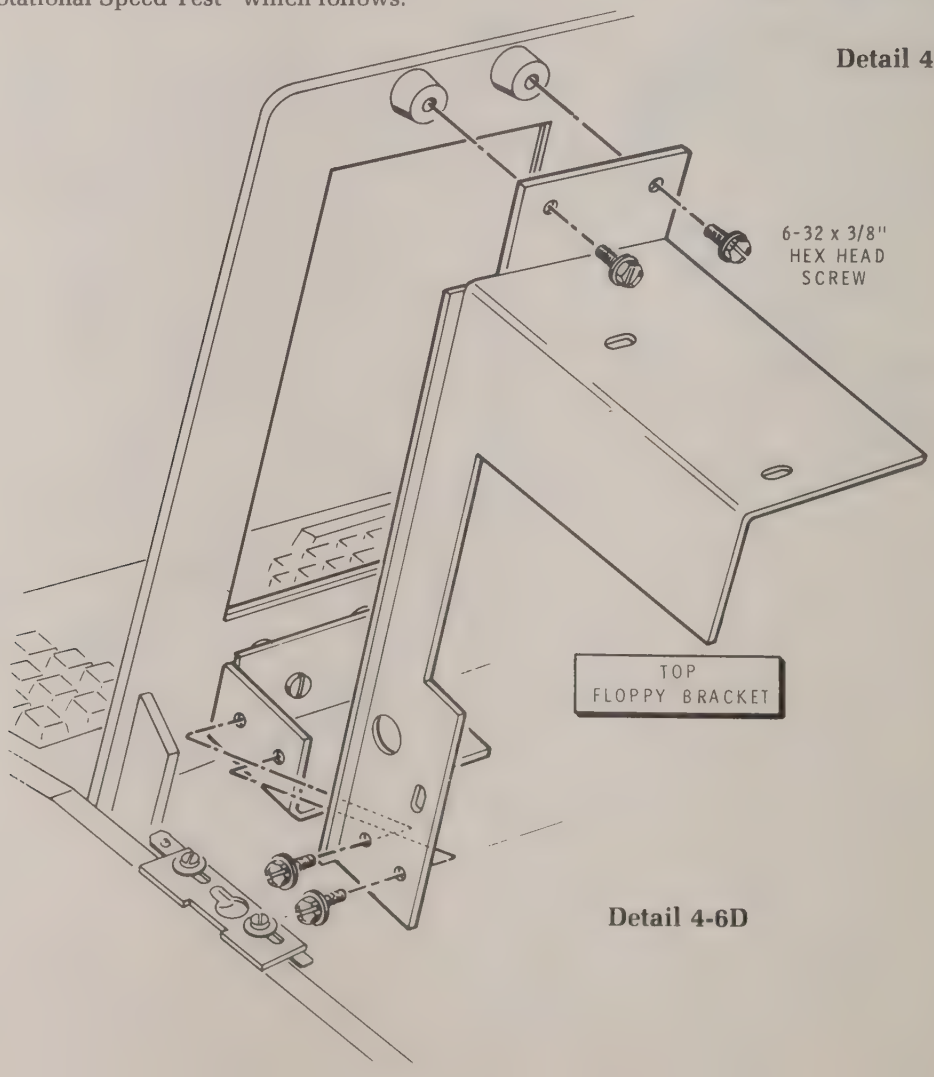
( ) Insert the floppy drive unit part way into the Computer as shown and connect the 4-wire power plug as shown in Pictorial 4-6.

( ) Connect the ribbon cable to the floppy as shown.

Leave the floppy drive unit in its present position and perform the "Rotational Speed Test" which follows.



Detail 4-6E



Detail 4-6D

Rotational Speed Test

This test will check the rotational speed of the floppy disk drive. After the test starts, it will display the disk drive speed test message and a "speed equals" message as follows:

Disk drive rotational speed test

Drive speed =

The word "Working" will be flashed on and off at the home position and a drive speed number (in octal), which should be as close to 200 as possible, will update with each flash on the screen. The rotational speed tolerance is one percent. The displayed value should be between 166 and 212 (remember, octal numbers jump from 177 to 200). Do not adjust the speed unless it is out of tolerance.

Your drive unit has a control labeled R30 or R7, which is used to vary the drive speed. This control is located near the lower edge of the circuit board as shown in Pictorial 4-7, or on the other side of the drive if your drive is like the one shown in Detail 4-7A. This adjustment may be extremely sensitive so, if an adjustment is necessary, do not turn it far in either direction. Less than one degree of rotation in either direction should bring the drive speed into tolerance. Turn the control clockwise to decrease the speed or counterclockwise to increase it.

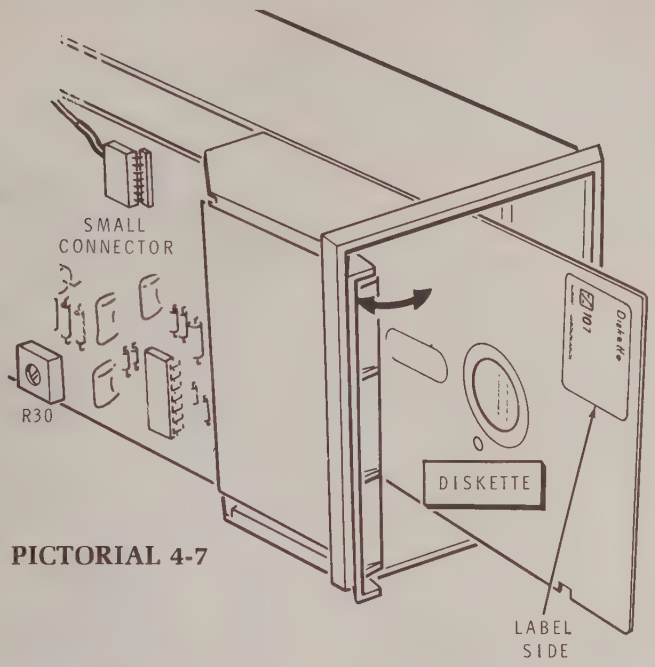
You may have to perform this test again at some future time, depending upon how heavily your floppy is used. As the drive bearings wear, the speed may change slightly. Fluctuations within the tolerance are normal and may be attributed to variations in temperature and humidity.

( ) Plug in the line cord and turn on the Computer.

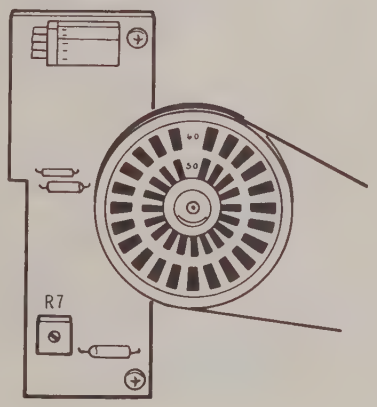
( ) Refer to Pictorial 4-7, install a diskette into the drive as shown, and close the door. The test will not start until this is done. Also, the test will **not** damage the data on the diskette.

( ) Be sure the OFF LINE key is up (out). Then obtain the "H:" prompt. (Simultaneously push the right-hand SHIFT and RESET keys if necessary.)

( ) Type G7372⏎. The total entry will be:
H: **Go 7372⏎**
and the test will start.



PICTORIAL 4-7



Detail 4-7A

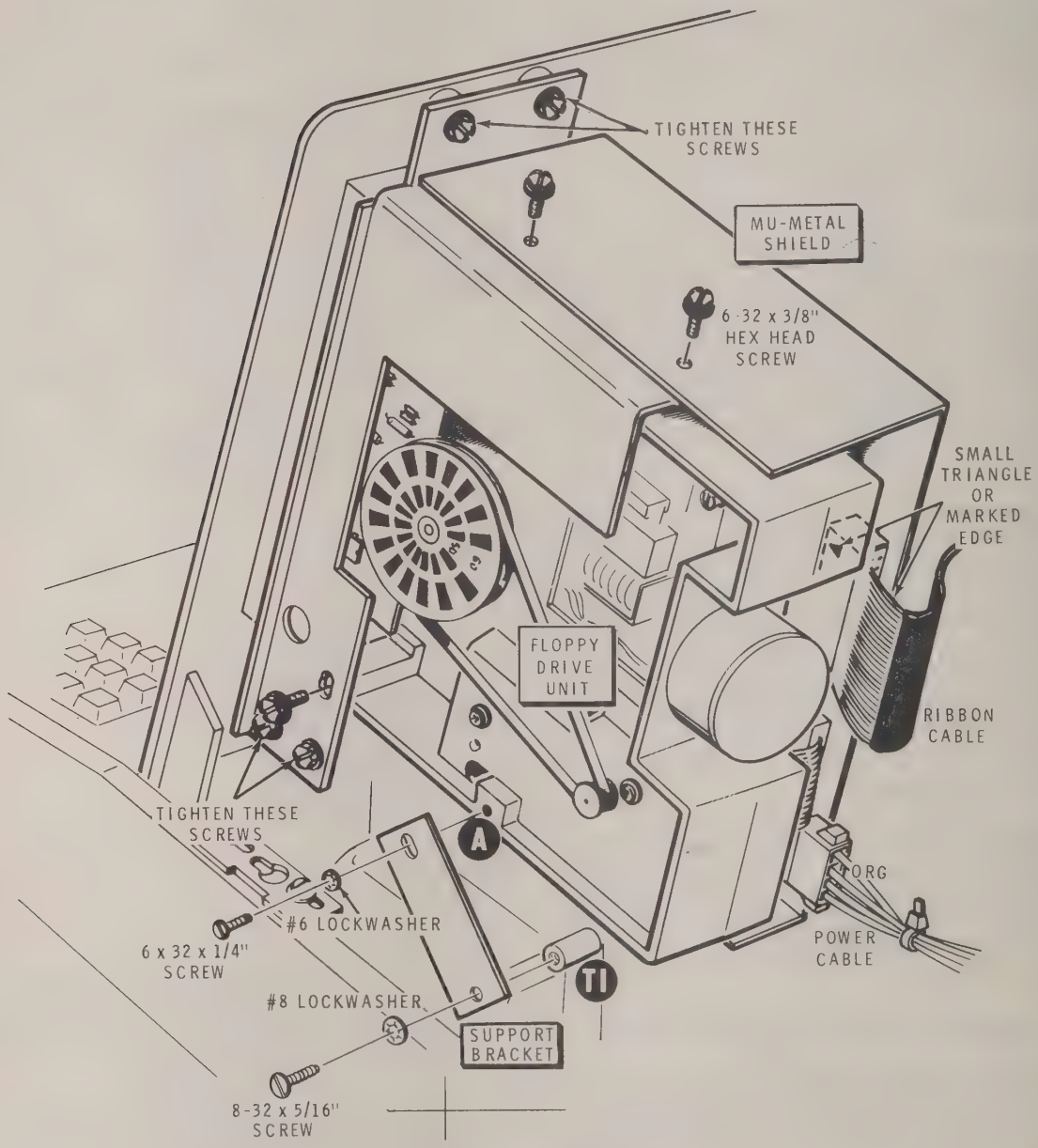
( ) If the drive speed is less than 166 or greater than 212, carefully adjust R30 or R7 with a small screwdriver until the speed is within tolerance and as close to 200 as possible.

( ) Simultaneously push the right-hand SHIFT and RESET keys to stop the test.

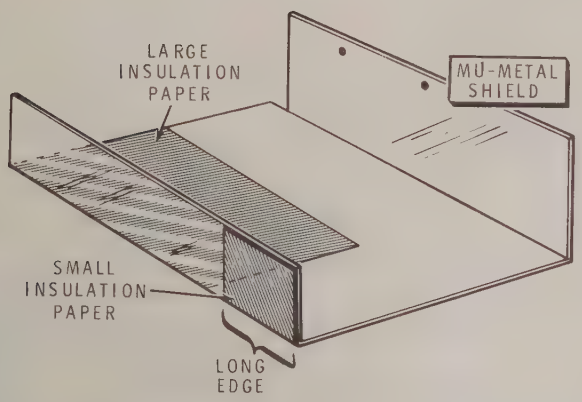
( ) Remove the diskette from the drive unit.

( ) Turn off the Computer and unplug the line cord.

( ) Unplug the floppy drive unit and remove it from the Computer. NOTE: Be sure the small connector on the floppy drive unit is still connected.



PICTORIAL 4-8



Detail 4-8A

Final Installation

Refer to Pictorial 4-8 for the following steps.

( ) Refer to Detail 4-8A and position the mu-metal shield as shown. Then remove the protective backing from the large insulation paper and press the insulation paper onto the shield as shown.

( ) Again refer to Detail 4-8A and, in a similar manner, install a small insulating paper on the mu-metal shield. Position the paper with the long edge as shown.

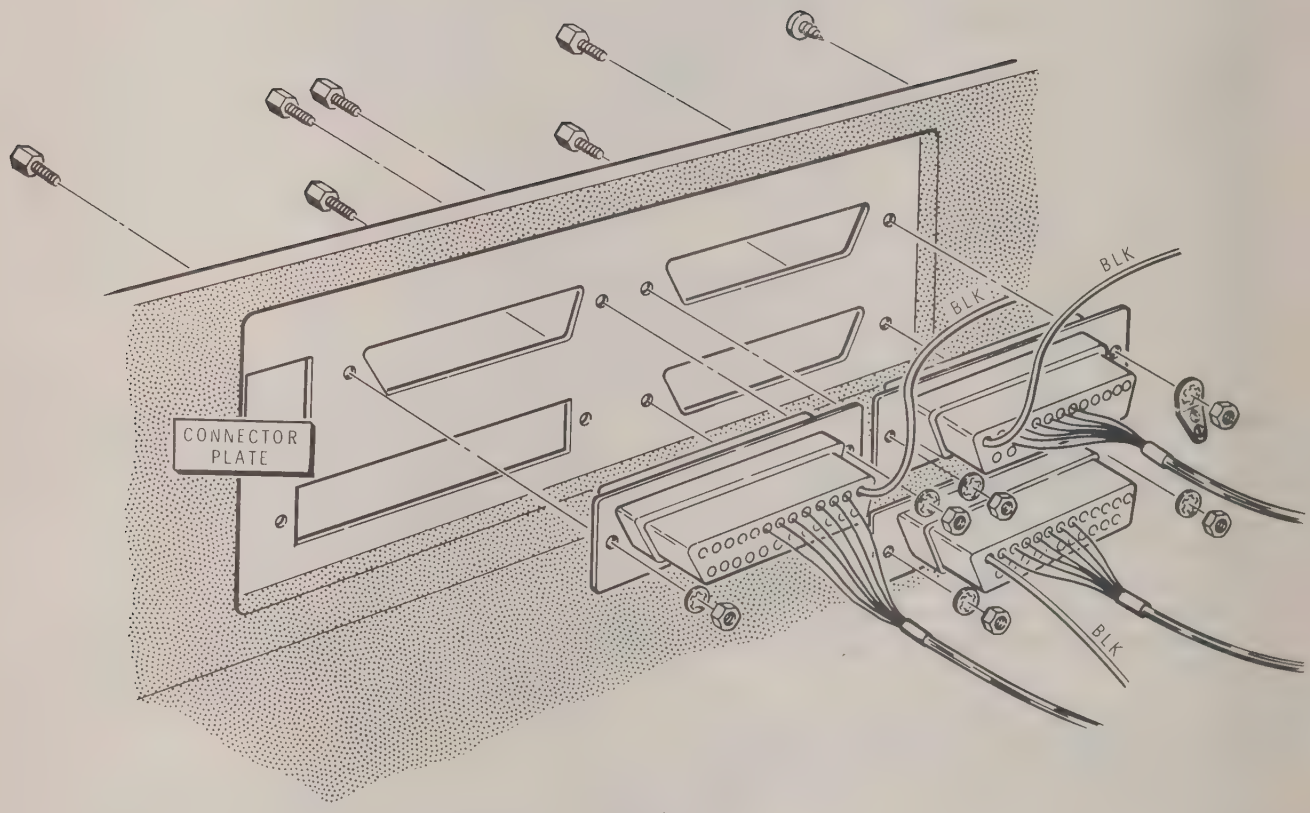
( ) Position the mu-metal shield into the Computer as shown. Then slide the floppy drive unit part way into the Computer and reconnect the power cable and ribbon cable.

( ) Slide the drive unit the rest of the way into the Computer and secure it and the Mu-metal shield in place with three 6-32 × 3/8″ hex head screws as shown.

( ) Tighten the four screws you left loose earlier.

( ) Install the support bracket as shown in Pictorial 4-8. Use an 8-32 × 5/16″ screw and #8 lockwasher at T1 and a 6-32 × 1/4″ screw and #6 lockwasher at A.

If you have another accessory to install, do it now. Otherwise, proceed to the "System Configuration" (Page 5-1) and "Operation" (Page 6-1) sections.



Detail 4-9A

H88-3 SERIAL INTERFACE

PARTS LIST

HEATH Part No.	QTY.	DESCRIPTION
181-3053	1	Wired serial interface circuit board
134-1070	1	DTE cable
134-1073	1	DCE cable

NOTE: IC sockets U603, U607, and U608 do not have IC's installed in them.

INSTALLATION

( ) Unplug the line cord and remove the cabinet shell. See Pictorial 3-1 on Page 3-2.

Refer to Pictorial 4-9 (Illustration booklet Page 5) for the following steps.

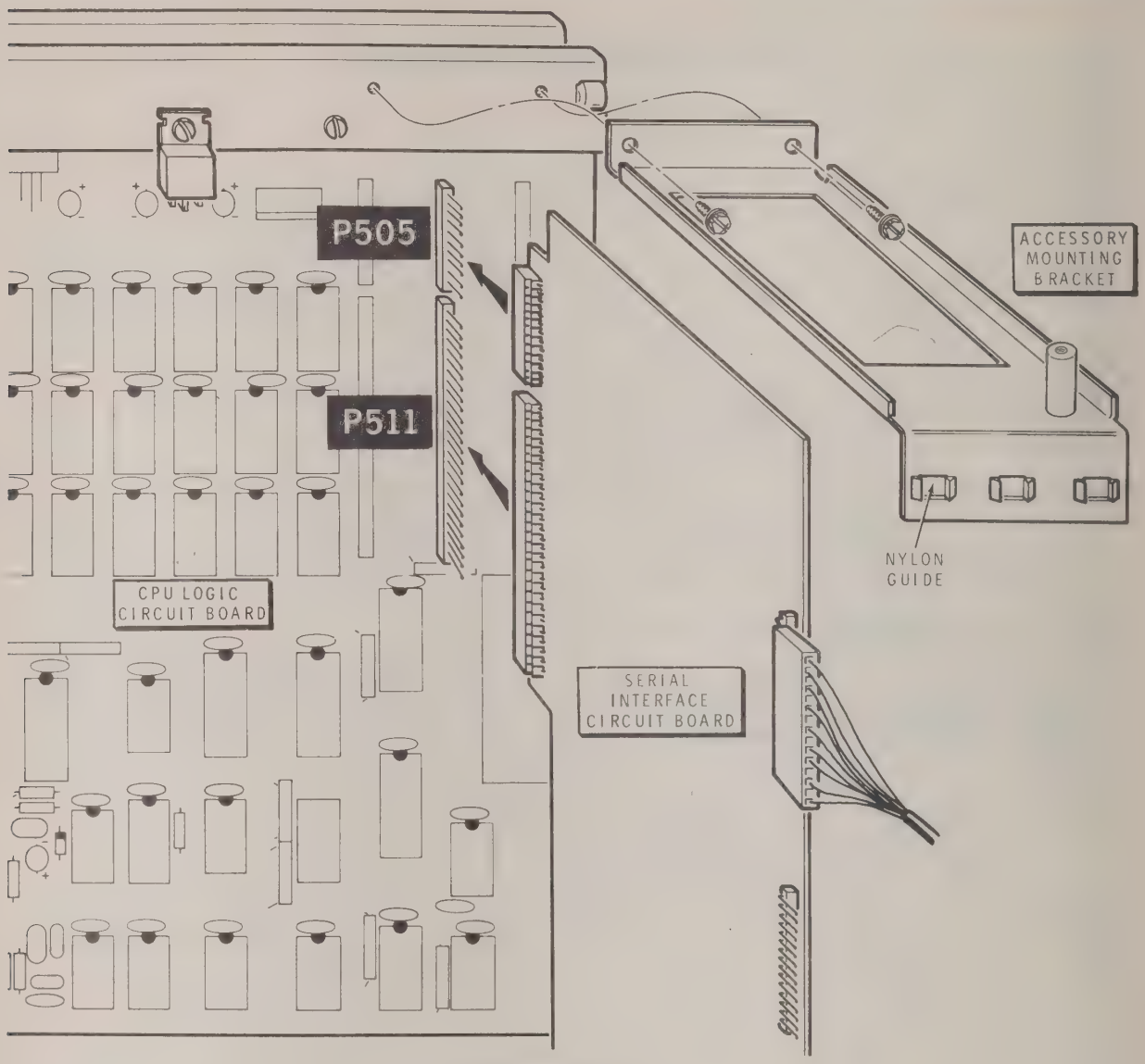
( ) Remove the four screws that hold the rear panel connector plate and set the screws aside.

( ) Refer to Detail 4-9A and mount the D-connector of the DTE cable (#134-1070) at B on the rear panel. Use the hardware that is already at location B and mount the connector so the black wire is positioned as shown in the Pictorial.

( ) Connect the black wire to solder lug C and solder the connection. There is another wire already connected here.

( ) Route the free end of the cable through the rear panel opening and under the terminal and CPU logic circuit boards, and plug it onto P605 of the serial interface circuit board. Be sure the brown wire is positioned as shown in the Pictorial.

( ) In a similar manner, install the DCE cable (#134-1073) at A, solder the black wire to solder lug C, route the cable under the circuit boards, and plug the free end into P603.



PICTORIAL 4-10

Refer to Pictorial 4-10 for the following steps.

( ) Remove the two screws that hold the right-hand accessory mounting bracket and set the bracket aside temporarily.

( ) Plug the serial interface circuit board into the CPU logic circuit board at P505 and P511.

( ) Remount the accessory mounting bracket. Be sure the nylon guide holds the edge of the circuit board.

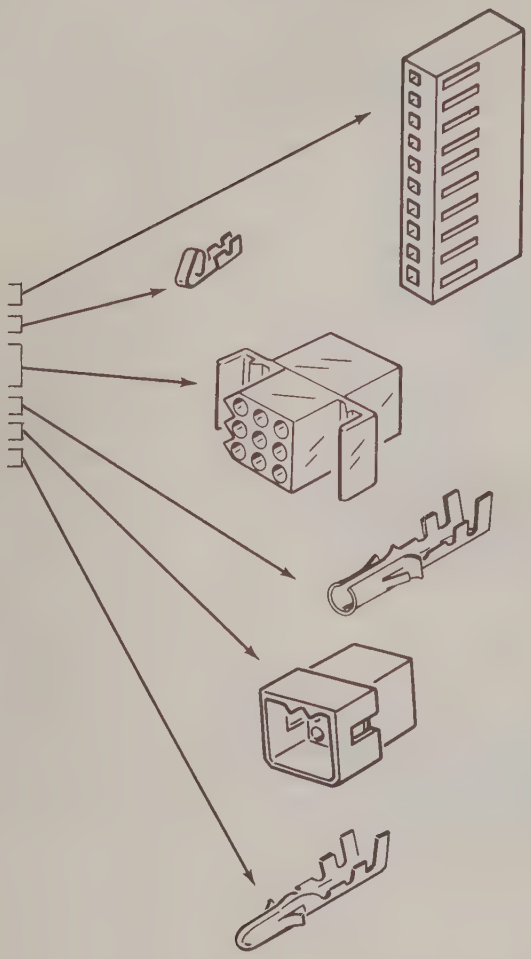
This completes the installation of your Serial Interface Accessory. Refer to "System Configuration" and the "Operation" section to properly set the circuit board programming jumpers and learn about the operation of this Accessory.

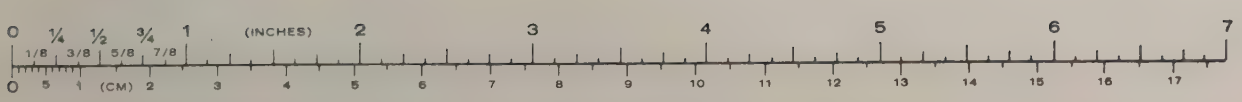
If you intend to install the H88-5 Cassette Interface accessory, proceed to Page 4-15 and do it now. Otherwise, replace the rear panel connector plate with the four screws you removed earlier. Be careful and do not overtighten these screws. Then proceed to "System Configuration" on Page 5-1.

H88-5 CASSETTE INTERFACE

PARTS LIST

HEATH Part No.	QTY.	DESCRIPTION
181-3054	1	Wired cassette interface circuit board
343-11	8′	Shielded cable
347-55	2′	Flat 8-wire cable
347-60	8′	2-wire cable
432-958	1	Flat connector shell
432-866	8	Spring connector
432-794	1	Male connector shell with ears
432-855	8	Female pin
432-183	1	Female connector shell
432-854	8	Male connector
438-26	2	Red phone plug
438-52	2	Black subminiature phone plug
346-67	2-1/2″	Heat-shrinkable tubing



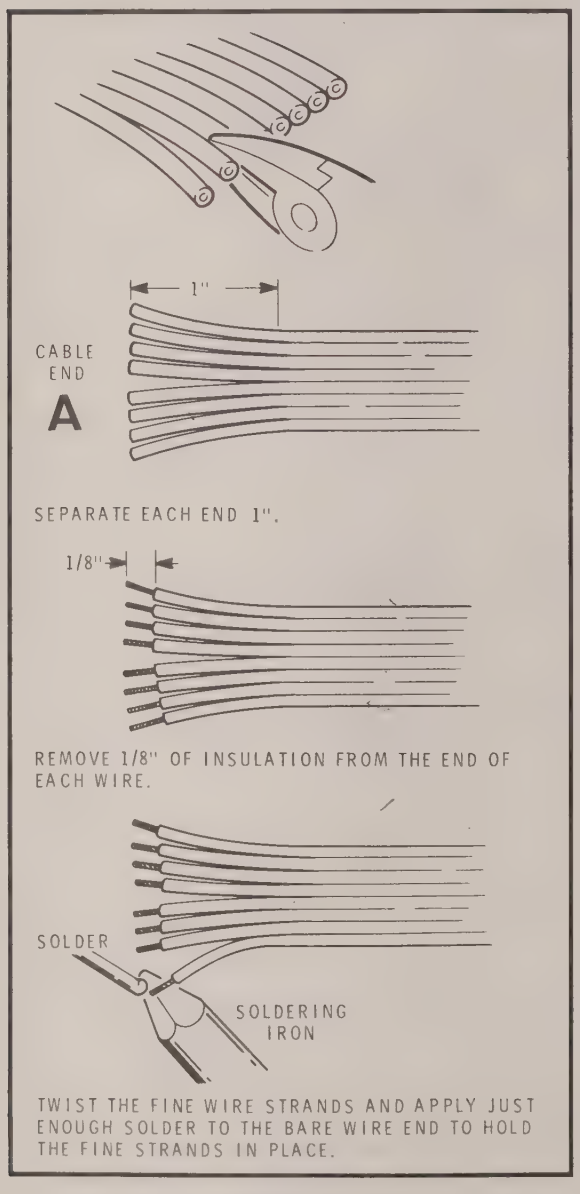


CABLE ASSEMBLY

Refer to Pictorial 4-11 (Illustration Booklet, Page 6) for the following steps.

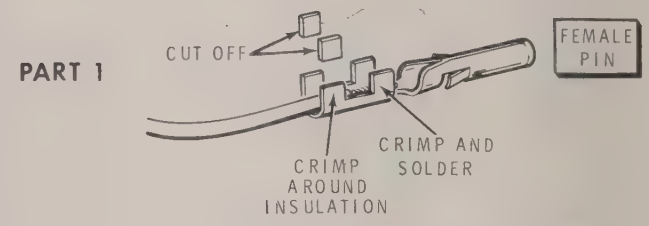
( ) Cut a 2-foot length of flat 8-wire cable

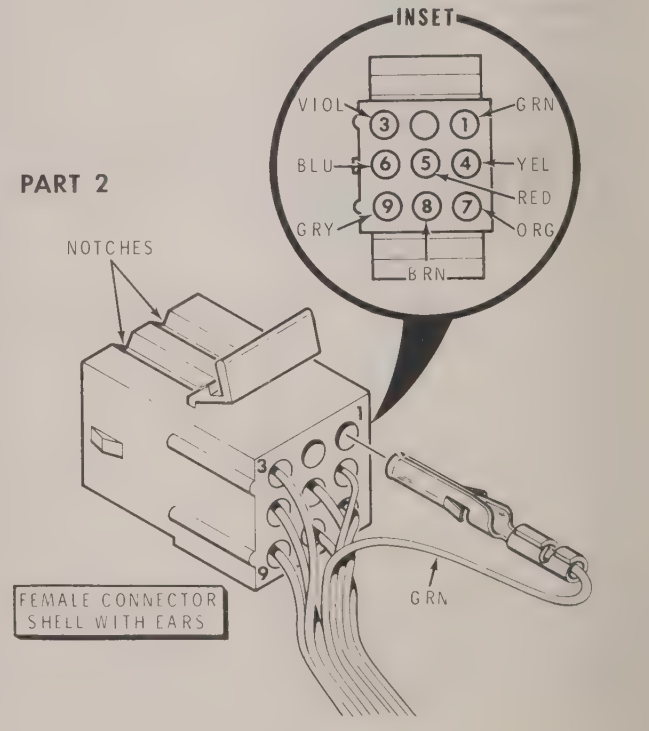
( ) Refer to Detail 4-11A and prepare end A of the flat 8-wire cable as shown. Use diagonal cutters or a knife to start separating the wires.


1"
CABLE END
A
SEPARATE EACH END 1".
1/8"
REMOVE 1/8" OF INSULATION FROM THE END OF EACH WIRE.
SOLDER
SOLDERING IRON
TWIST THE FINE WIRE STRANDS AND APPLY JUST ENOUGH SOLDER TO THE BARE WIRE END TO HOLD THE FINE STRANDS IN PLACE.

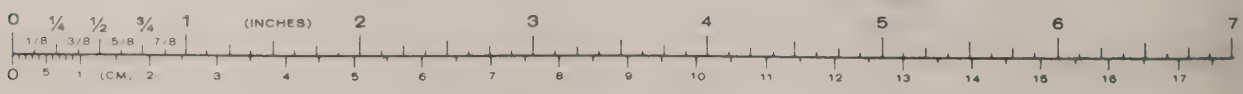
Detail 4-11A

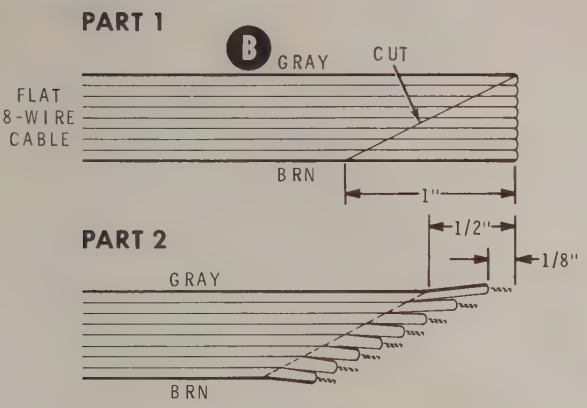
( ) Refer to Part 1 of Detail 4-11B and crimp and solder a female pin (#432-855) on the end of each prepared wire. Remove 1/2 of each indicated tab on the female pin before you crimp it to the wire.





Detail 4-11B





Detail 4-11C

NOTE: Refer to Part 2 of Detail 4-11B and insert the female pins on the end of the following wires into the male connector shell with ears as follows. Note that the hole numbers are stamped in the back of the connector. Each time you install a pin, gently pull on the wire to make sure the wire is locked in place.

( ) Green wire in hole 1.

( ) Do not install a pin in hole 2.

( ) Violet wire in hole 3.

( ) Yellow wire in hole 4.

( ) Red wire in hole 5.

( ) Blue wire in hole 6.

( ) Orange wire in hole 7.

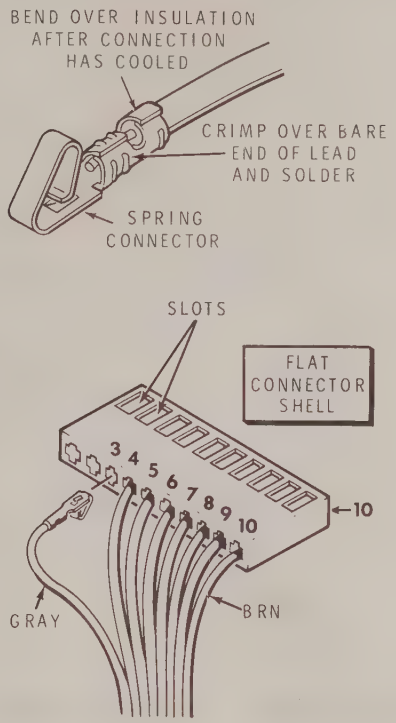
( ) Brown wire in hole 8.

( ) Gray wire in hole 9.

( ) Refer to Part 1 of Detail 4-11C and cut end B of the flat 8-wire cable as shown.

( ) Refer to Part 2 of Detail 4-11C, separate each wire end 1/2", remove 1/4" of insulation from the wire ends, apply a small amount of solder to the bare ends, and then cut the bare ends to 1/8".

( ) Refer to Detail 4-11D and crimp and solder a spring connector (#432-866) on the end of each prepared wire.



Detail 4-11D

NOTE: Insert the spring connector on the ends of the following wires into the flat connector shell as follows. Note that the number 10 is stamped in the end of the connector. Each time you install a connector pin, gently pull on the wire to make sure the wire is locked in place.

( ) Gray wire in hole 3.

( ) Violet wire in hole 4.

( ) Blue wire in hole 5.

( ) Green wire in hole 6.

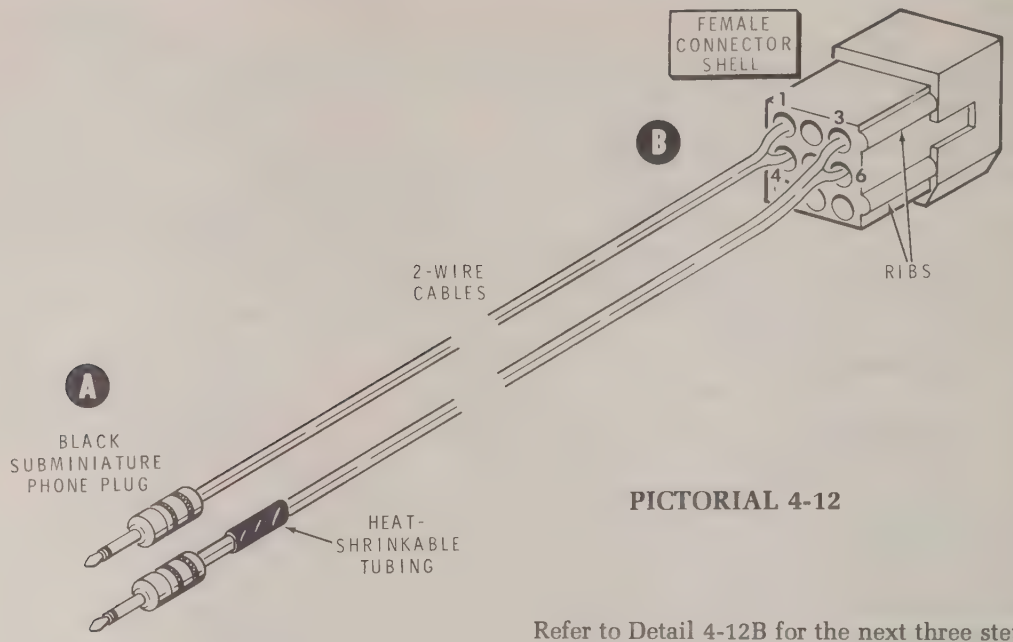
( ) Yellow wire in hole 7.

( ) Orange wire in hole 8.

( ) Red wire in hole 9.

( ) Brown wire in hole 10.

Set the cable aside.



PICTORIAL 4-12

Refer to Pictorial 4-12 for the following steps.

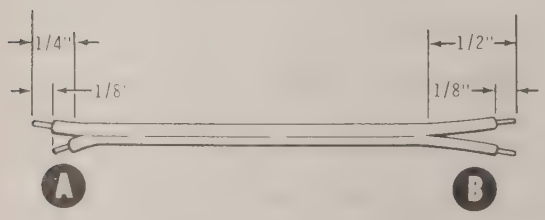
( ) Cut the heat-shrinkable tubing into two equal lengths.

( ) Cut the shielded cable into two equal lengths.

( ) Cut the 2-wire cable into two equal lengths.

NOTE: In the following steps, you will assemble two 2-wire cables. Complete the first cable; then repeat the steps where indicated by double check spaces for the second cable. Some steps use only one check space. Perform these steps only once as instructed.

( ) ( ) Refer to Detail 4-12A and prepare ends A and B of the 2-wire cable as shown.



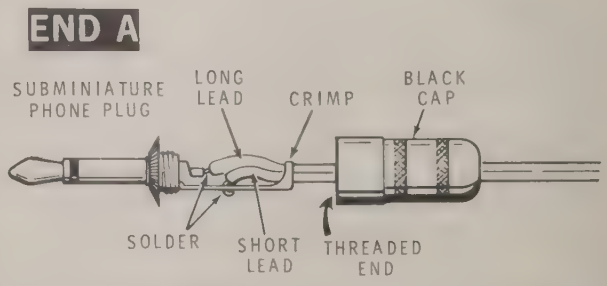
Detail 4-12A

Refer to Detail 4-12B for the next three steps.

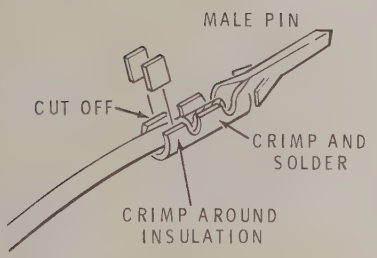
( ) ( ) Remove the black cap from the subminiature phone plug. Slide the cap over the 2-wire cable at end A with the threaded end as shown.

( ) ( ) Solder the long and short wires at end A of the 2-wire cable to the indicated lugs on the subminiature phone plug. NOTE: You may find it easier to clamp the phone plug in a small vice to hold it steady while you solder the wires to it.

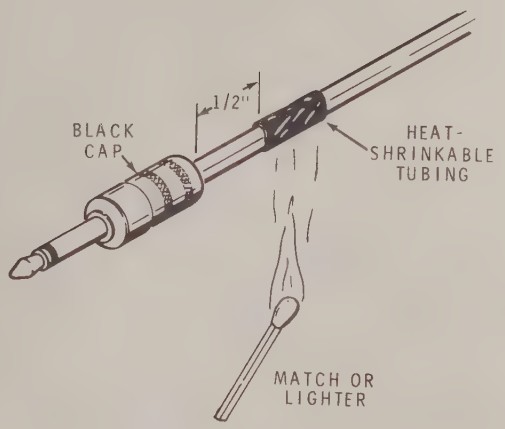
( ) ( ) After the connections cool, crimp the indicated lugs around the 2-wire cable as shown. Replace the black cap on the phone plug.



Detail 4-12B



Detail 4-12C



Detail 4-12D

( ) ( ) Refer to Detail 4-12C and crimp and solder a male pin on the end of each prepared wire at end B of the 2-wire cable.

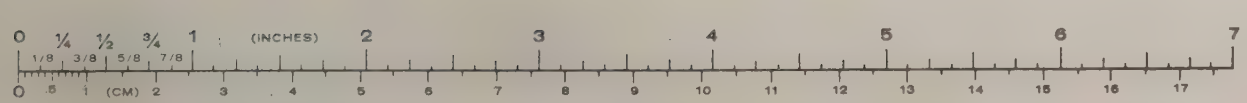
( ) Slide a length of heat-shrinkable sleeving onto end A of the remaining 2-wire cable. Then repeat the previous steps and assemble the second 2-wire cable.

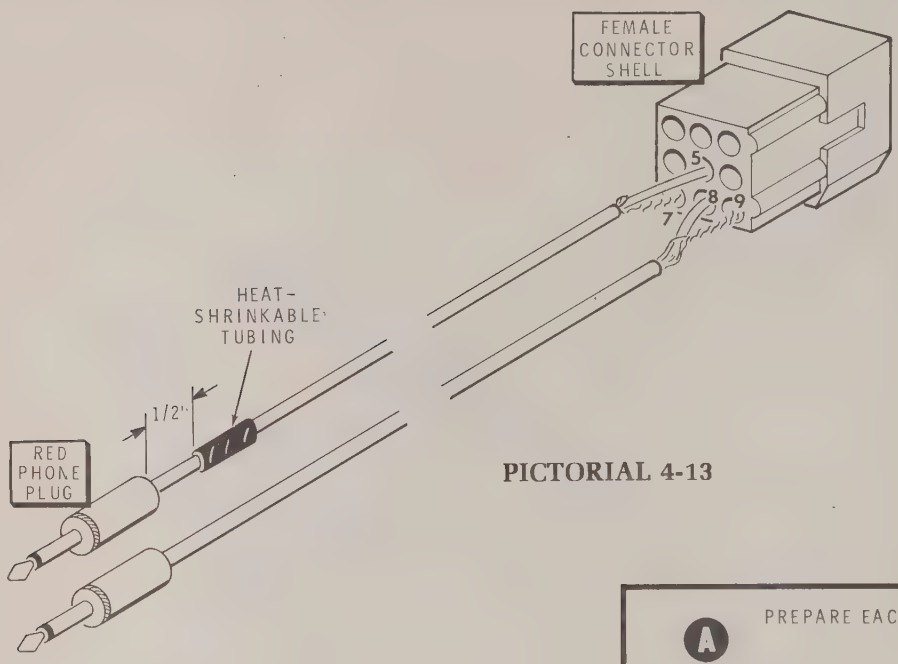
( ) Position the female connector shell with the ribs as shown in Pictorial 4-12.

( ) Push the male pins of the 2-wire cable with heat-shrinkable tubing into connector holes 3 and 6 (either pin into either hole). Gently pull on the wires to make sure they are locked in place.

( ) Position the length of heat-shrinkable tubing approximately 1/2" from the subminiature phone plug. Shrink the tubing in place with a lighter or match. Be careful not to melt the cable insulation.

( ) Push the male pins of the remaining 2-wire cable into connector holes 1 and 4 (either pin into either hole). Gently pull on the wires to make sure they are locked in place.





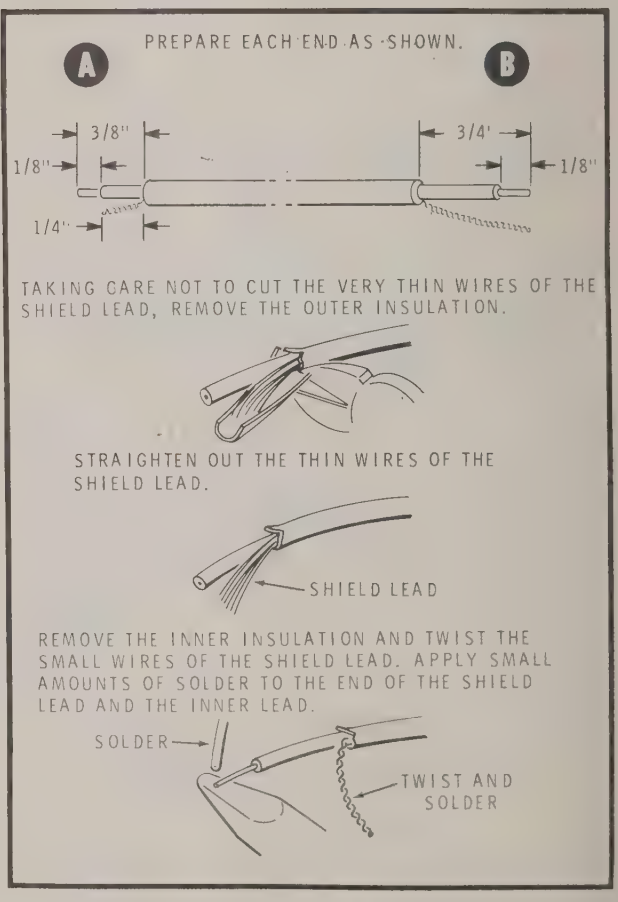
PICTORIAL 4-13

Refer to Pictorial 4-13 for the following steps.

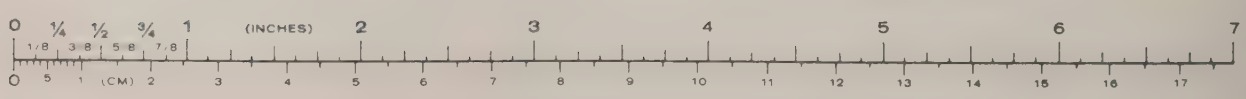
NOTE: In the following steps, you will be instructed to assemble two shielded cables. After you complete the first cable, repeat the steps for the second cable; double check spaces are provided.

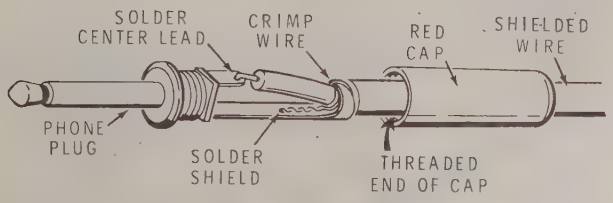
( ) ( ) Refer to Detail 4-13A and prepare a shielded cable as shown.

( ) ( ) Remove the red cap on one of the phone plugs and slide it over the shielded cable at end A with the threaded end as shown in Detail 4-13B.


PREPARE EACH END AS SHOWN.
A
B
3/8"
1/8"
3/4"
1/8"
1/4"
TAKING CARE NOT TO CUT THE VERY THIN WIRES OF THE SHIELD LEAD, REMOVE THE OUTER INSULATION.
STRAIGHTEN OUT THE THIN WIRES OF THE SHIELD LEAD.
SHIELD LEAD
REMOVE THE INNER INSULATION AND TWIST THE SMALL WIRES OF THE SHIELD LEAD. APPLY SMALL AMOUNTS OF SOLDER TO THE END OF THE SHIELD LEAD AND THE INNER LEAD.
SOLDER
TWIST AND SOLDER

Detail 4-13A

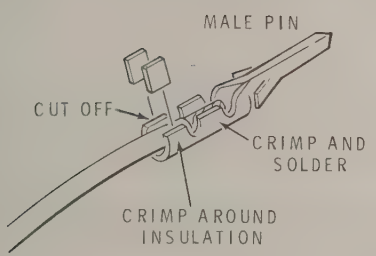




Detail 4-13B

( ) ( ) Refer to Detail 4-13B and solder the center lead of the shielded cable to the indicated lug of the phone plug. Solder the shield to the indicated location on the phone plug. After the connection cools, crimp the tabs around the wire and replace the red cap on the phone plug.

( ) ( ) Refer to Detail 4-13C and crimp and solder a male pin on the end of the center lead and the shield lead of end B of a shielded cable.



Detail 4-13C

( ) Slide a length of heat-shrinkable tubing over end A of the second shielded cable and repeat the previous steps and assemble another shielded cable.

( ) Push the male pins of the shielded cable with heat-shrinkable tubing into the female connector shell holes as follows. Gently pull on the wires to make sure they are locked in place.

Shield lead to hole 7.

Center lead to hole 5.

( ) Position the length of heat-shrinkable tubing approximately 1/2″ from the phone plug. Shrink the tubing in place with a lighter or match.

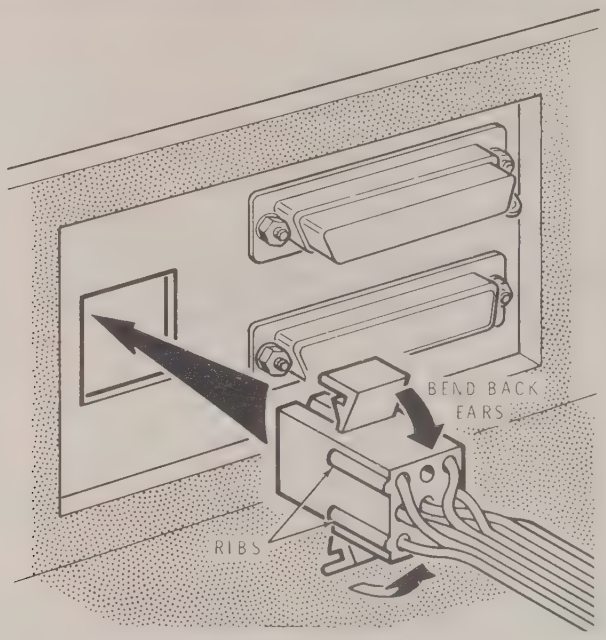
( ) Push the male pins of the remaining cable into the connector holes as follows. Gently pull on the wires to make sure they are locked in place.

Shield lead to hole 9.

Center lead to hole 8.

( ) If an ohmmeter is available, make continuity checks on the cables to be sure there are no short or open circuits.

( ) Set this cable assembly aside.



Detail 4-14A

INSTALLATION

( ) Unplug the line cord and remove the cabinet shell. See Pictorial 3-1.

Refer to Pictorial 4-14 (Illustration Booklet, Page 7) for the following steps.

( ) If it has not already been done, remove the four screws that hold the rear panel connector plate and set the screws aside.

( ) Refer to Detail 4-14A and mount the male connector shell with ears in the rear panel at D as shown.

( ) Plug the prepared cable assembly into this connector at D.

( ) Route the flat 8-wire cable through the rear panel opening and under the terminal logic and CPU logic circuit boards. (You may remove the circuit boards if necessary.)

( ) Remove the two screws that hold the right-hand accessory mounting bracket and set the bracket aside temporarily.

( ) Plug the cassette interface circuit board into the CPU logic circuit board at P504 and P510.

( ) Remount the accessory mounting bracket. Be sure the nylon guide holds the edge of the circuit board.

( ) Reinstall the CPU logic circuit board and reconnect the cables that you disconnected earlier.

( ) Plug the free end of the flat 8-wire cable into plug P703 of the cassette interface circuit board. Be sure the gray wire is positioned up as shown.

( ) Replace the rear panel connector plate with the four screws you removed earlier. Do not overtighten these screws.

This completes the installation of your Cassette Interface Accessory. Proceed to "System Configuration" and the "Operation" section to properly set the circuit board programming jumpers and learn about the operation of this Accessory.

SYSTEM CONFIGURATION

If you are using your Computer with Heath peripherals, proceed to the "Heath System Configuration" section that follows. However, if you are using your Computer with non-Heath peripherals, proceed to the "Non-Heath System Configuration" information on Page 5-7.

HEATH SYSTEM CONFIGURATION

This section of the Manual shows you the normal switch and jumper positions. If you like, check them to be sure they are in their proper positions, or set them (as explained) for the operation that you desire.

Your Computer communicates with your peripherals at RS-232C signal levels. 25-pin "D" connectors on the rear panel conform to RS-232C standards and mate with most equipment that conforms to this standard.

Refer to the following sections that pertain to your Computer.

Terminal Logic Circuit Board

Carefully tilt back the cabinet shell. See Pictorial 3-1.

Be sure the POWER switch is off. Then remove the two screws that hold the top of the terminal logic circuit board (the rear circuit board). Remove the two screws that hold the top of the CPU logic circuit board. Disconnect the cables, as necessary, and lift the circuit boards up out of the Computer.

SWITCH S402

Push all of the switches on S402 up (0) as shown in Pictorial 5-1 (Illustration Booklet, Page 8).

If you ever want to change these switch positions, they are defined as follows:

SECTION	DESCRIPTION
0	0 = underscore cursor; 1 = block cursor
1	0 = key click; 1 = no key click
2	0 = discard past end of line; 1 = wrap around
3	0 = no auto LF on CR; 1 = auto LF on CR
4	0 = no auto CR on LF; 1 = auto CR on LF
5	0 = Heath mode; 1 = ANSI mode
6	0 = keypad normal; 1 = keypad shifted
7	0 = 60 Hz refresh; 1 = 50 Hz refresh

SWITCH S401

Refer to Pictorial 5-1 for the following steps.

This switch (located on the terminal logic circuit board) sets the following power-up and reset modes:

SWITCH SECTION	DESCRIPTION
0-3	Baud Rate
4	Parity Enable
5	Odd/Even Parity
6	Normal/Stick Parity
7	Half/Full Duplex

The particular configuration that you select is initialized when you power-up the Computer or when you perform a Computer Reset. Pictorial 5-1 shows the location of switch S401. Remember that, as you look at switch S401 from the front of the Computer, you select the one (1) positions of the switch by pushing the switches down, and you select the zero (0) positions by pushing the switches up.

( ) Set switch S401 for:

MODE	SWITCH SECTION 0	1	2	3	4	5	6	7
9600 Baud	0	0	1	1				
No Parity					0			
Even Parity						0		
Normal Parity							0	
Full Duplex								1

When no parity is selected, you can set the even and normal parity switch sections to either position since they will be ignored.

Each function of switch S401 is explained in the following text.

Baud Rate

When used as a Computer, the baud rate must be set to 9600. If you use your Computer as a terminal (see Page 6-18), you can reset the baud rate as explained below.

You can select any of 12 different baud rates (110-9600). To do this, place sections 0, 1, 2, and 3 of switch S401 to the proper positions as shown below. The baud rate will be initialized (or updated) upon Reset or during power-up.

BAUD RATE	SWITCH SECTION 0	1	2	3
N/A	0	0	0	0
110	1	0	0	0
150	0	1	0	0
300	1	1	0	0
600	0	0	1	0
1200	1	0	1	0
1800	0	1	1	0
2000	1	1	1	0
2400	0	0	0	1
3600	1	0	0	1
4800	0	1	0	1
7200	1	1	0	1
9600	0	0	1	1

Parity

You can program the ACE (Asynchronous Communication Element) to either generate or eliminate the parity bit. Section 4 of switch S401 selects the parity bit.

Down (1) = Parity
Up (0) = No Parity

Heath Software does not check parity.

Odd/Even Parity

If section 4=1, then section 5 of switch S401 selects odd or even parity.

Down (1) = Even Parity
Up (0) = Odd Parity

Normal/Stick Parity

If section 4=1, then section 6 of switch S401 sets the ACE to transmit and receive either stick or normal parity.

Down (1) = Stick Parity
Up (0) = Normal

Half/Full Duplex

Section 7 of switch S401 selects either full or half duplex communications between the computer and the terminal sections.

Down (1) = Full Duplex
Up (0) = Half Duplex

Heath Software supports full duplex operation. Set section 7 to 0 for full duplex operation.

( ) Replace the circuit boards and reconnect their cables.

CPU Logic Circuit Board

Refer to Pictorial 5-2 (Illustration Booklet, Page 9) for the following steps.

SWITCH SW501

( ) Set section 5 of the switch to ON.

( ) Set the other seven switch sections to OFF.

The sections of switch SW501 are defined as follows:

SECTION	DESCRIPTION
0	Not used; leave at 0.
1	Not used; leave at 0.
2	Not used; leave at 0.
3	Not used; leave at 0.
4	Not used; leave at 0.
5	0 = Performs RAM memory test at reset. Must set to 1 to stop. 1 = Normal.
6, 7	0 = Sets the baud rate to 9600 when both switches are at 0. Other positions are undefined.

PROGRAMMING JUMPERS

The positions of jumpers JJ501 through JJ504 are determined by the amount of memory you have installed. Position the jumpers according to the following chart. If necessary, see "Accessory Installation" "H88-2 Memory Expansion" on Page 4-1.

	MEMORY	JJ504	JJ503	JJ502	JJ501
( )	16k bytes	B	0	0	0
( )	32k bytes	B	0	0	1
( )	48k bytes	B	0	1	0

The positions of programming jumpers JJ505 through JJ508 are determined by the types and locations of the system PROM's. Be sure the jumpers are installed as follows:

	JUMPER	LOCATION
( )	JJ506	0
( )	JJ507	0
( )	JJ505	0
( )	JJ508	B

Floppy Disk (H88-1)

The floppy disk interface circuit board must be installed only at P506 and P512 on the CPU logic circuit board. See Pictorial 4-5 for programming the floppy drive unit. (NOTE: If you purchased the H89 Computer, the drive is already properly programmed.)

Serial Interface (HA88-3)

( ) Refer to Pictorial 5-3 (Illustration Booklet, Page 9) and set all three programming jumpers to off if this has not already been done.

These jumpers determine the interrupt priority of the ports. When a jumper is in the "OFF" position, no interrupt exists for that port. By installing the jumper at 3, 4, or 5, a restart 30, 40, or 50 (respectively) instruction is executed when the interrupt for that port occurs. If the ACE is set to generate interrupts, all interrupts are vectored through MTR88 and UIVEC (see the monitor 88 section of your software manual).

The first port on the circuit board is located at address 340/347 and is used by HDOS (Heath Disk Operating System) as the line printer port. However, it is not restricted to that use only, since it is a standard RS-232C interface with a "DCE" connector.

The second port on the circuit board is located at address 320/327 and is used by HDOS as the alternate terminal port. Again, it is a standard RD-2320 interface equipped with a "DCE" connector.

The third port on the circuit board is located at address 330/337 and is used as a general port. It is a standard RS-232C interface with a "DTE" connector (suitable for a MODEM). However, the system will not automatically configure the ACE at power up. It is up to you to initialize the ACE and install a handler in your system.

For programming the ACE, refer to Page 14-1 in the "Appendix."

Cassette Interface (H88-5)

Refer to Pictorial 5-4 for the following steps.

( ) JP701/JP702 — If you will be using only one tape recorder for both record and playback, then install jumpers at JP701 and JP702.

— If you will be using two recorders, one for recording and one for playback, then **do not** install jumpers at JP701 and JP702 but store the jumpers on the storage pins.

( ) JP703 — Place this jumper in the "box." This is the normal operating position.

With the jumper in the GND position (see the inset drawing), a continuous "0" tone (1200 Hz) is output to the audio output pin.

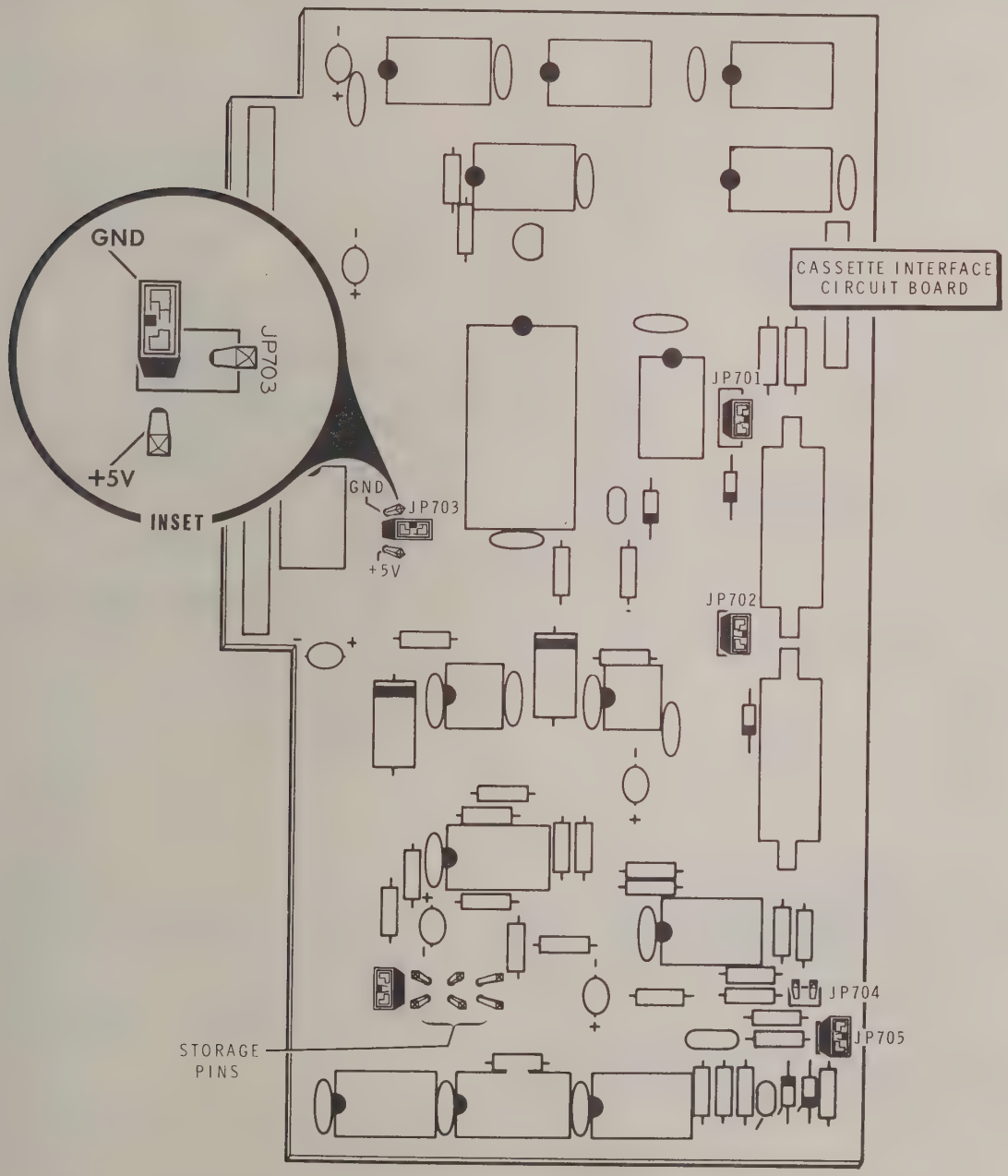
With the jumper in the + 5 V position, a continuous "1" tone (2400 Hz) is output to the audio output pin.

( ) JP704 — Do not install a jumper here unless you want to reduce the incoming signal level from approximately 500 mV peak-to-peak to 1 mV peak-to-peak. Normally, store this jumper on the storage pins.

( ) JP705 — Install a jumper here for normal operation. The jumper can be removed for troubleshooting purposes.

( ) Replace any circuit boards you removed, reconnect their cables, tighten all loose hardware, and close the cabinet shell.

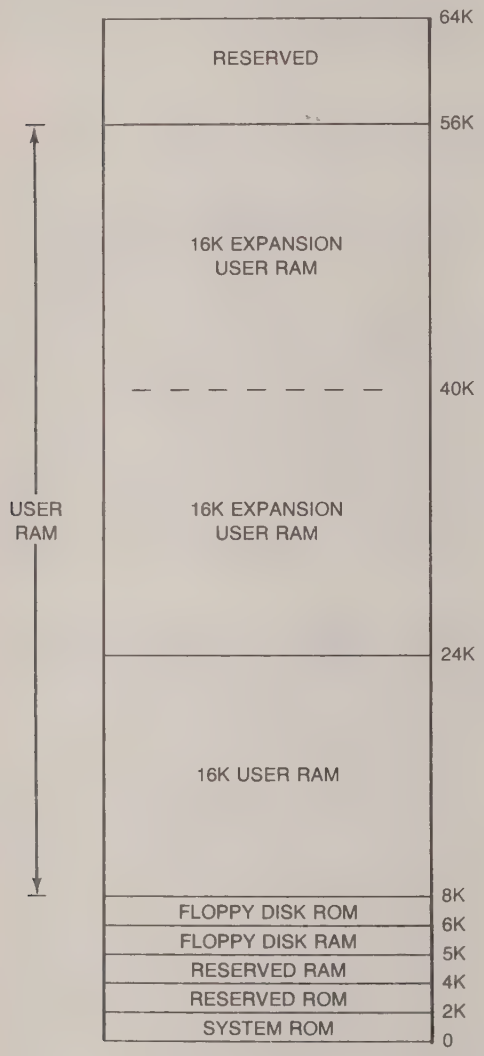
Proceed to the "Operation" section on Page 6-1.



PICTORIAL 5-4

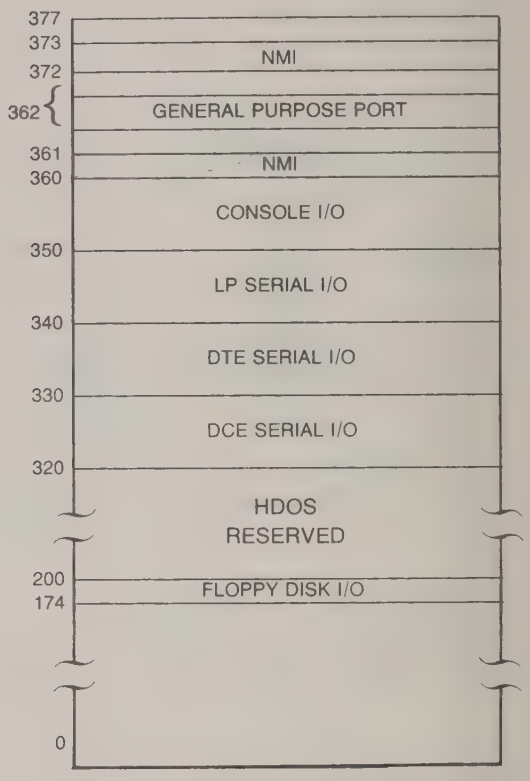
Memory Map

The Memory Map illustrates the use of the specified memory locations.

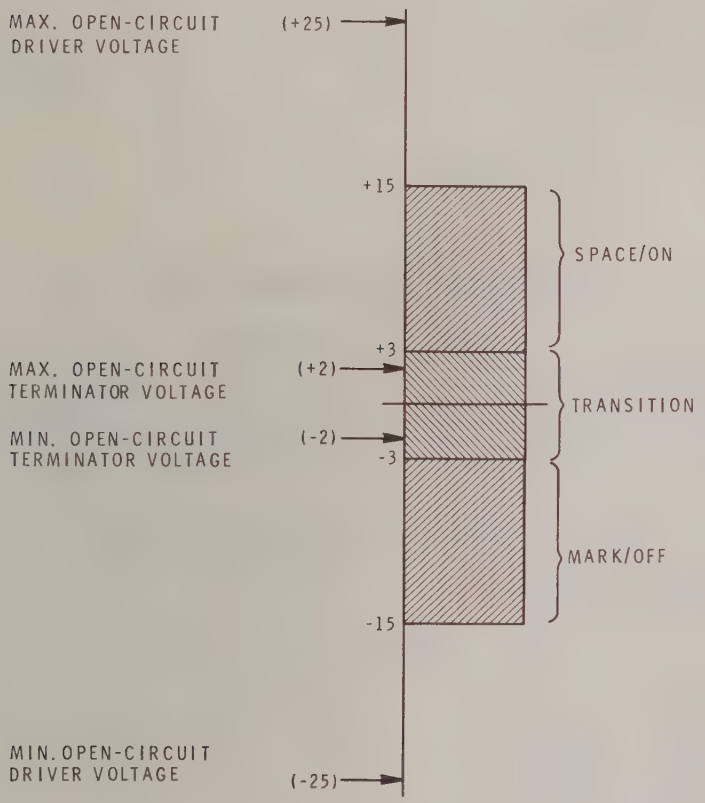


I/O Port Map

The I/O Port Map illustrates the use of the specified I/O port addresses.



NON-HEATH SYSTEM CONFIGURATION



PICTORIAL 5-5

If your Computer has the H88-3 Serial Interface Accessory installed, it is designed to interface with accessories that use the RS-232C standards of the Electronic Industries Association (EIA). This standard defines an asynchronous serial interface, its voltages (see Pictorial 5-5), its impedances, and its physical connectors.

RS-232C places all equipment into one of two general categories.

DTE — Data Terminal Equipment

DCE — Data Communication Equipment

Computers and MODEMS are two types of DCE; while terminals, printers, and most peripherals are DTE. Always connect a DTE to a DCE. Never connect two like types together.

Review the "Heath System Configuration" information to properly configure your computer and accessory circuit boards to operate with your peripherals.

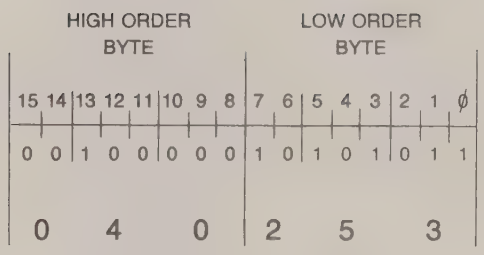

HIGH ORDER BYTE
LOW ORDER BYTE
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 ϕ
0 0 1 0 0 0 0 0 1 0 1 0 1 0 1 1
0 4 0 2 5 3

Figure 6-1

OPERATION

COMMAND SUMMARY

This section of the Manual is intended for those who are experienced in using computers. You should be familiar with BASIC programming and the Heath Disk Operating System (HDOS; if the H88-1 Floppy Disk Accessory is installed) before you proceed.

This summary is a list of the commands that your computer will respond to and the responses that it will make. You may enter commands in either upper or lower case characters. In all cases, pushing the DELETE or RUBOUT key prior to pushing the RETURN key (which terminates the command) will cancel the current command and cause the Computer to respond with the prompt (H:). All byte entries are in octal, and all address entries are in split octal. NOTE: Split octal is two 3-bit bytes as shown in Figure 6-1.

1. Boot HDOS:

   After you see the prompt (H:), enter "B". The Computer will complete the "Boot" message and then wait for a Carriage Return. Then it will transfer control to the BOOT ROM at 030000.

   Insert a diskette in the floppy disk drive and close the door.

   Comply with the on-screen message "Type spaces to determine baud rate," and type several spaces. Then press the RETURN key and enter the date. Then, once again press the RETURN key.

2. Load cassette tape:

   Place the cassette to be read in the recorder and press the PLAY button.

   After you see the prompt, enter "L". The monitor will complete the "Load" message and wait for a Carriage Return. The recorder will then start and the address into which the data is being loaded will be continuously displayed on the screen.

3. Dump cassette tape:

   NOTE: Prior to dumping the first copy of an original cassette tape, the user Program Counter value must be set to the program entry point (see 4 on the next page and below).

   Place the cassette in the recorder with the leader past the recording head. Then set the recorder to RECORD, and enter "D" following the prompt (H:). The Computer will complete the dump message.

At this point you have two options:

A. If a cassette has been previously loaded or dumped and a copy of the same memory locations and the same program entry point is to be made, simply enter a Carriage Return.

B. If an original tape is to be made, note that the user program counter must have already been set up using the "P" command below. At this point, enter the first address of memory to be dumped in split octal notation. Then enter a hyphen (-) and the last memory address to be dumped. To start the dump, enter a Carriage Return.

After you enter the Carriage Return, in step A or B above, the user program counter will be displayed during the time the cassette motor is allowed to come up to speed. Once the motor is up to speed, the address of the data being currently dumped will be continuously displayed.

4. Go to user routine:

After the prompt, enter "G". The Computer will complete the "Go" message and wait for either a Carriage Return or a new address and a Carriage Return. If you entered only a Carriage Return, control will be given to the routine at the address specified by the user program counter (such as after loading a cassette tape when the user program counter is set to the address specified by the tape). However, if you entered a new address and a Carriage Return, control will be given to the address you specified.

5. Set user Program Counter value:

After the prompt, enter "P". The Computer will complete the "Program Counter" message. Now you may enter a new program counter value (in split octal) and terminate it with a Carriage Return. If no new value is entered and you enter a Carriage Return, the current contents of the user Program Counter will be displayed. Again, a new value may be entered and terminated with a Carriage Return, or just entering a Carriage Return will cause a return to the prompt and the current value to be unaltered.

6. Substitute memory:

After the prompt, enter "S". The Computer will complete the "Substitute" message and wait for an address to be specified (in split octal notation) and ended with a Carriage Return. The address specified will then be displayed, followed by its contents. At this point, the contents may be changed by entering the octal value to be placed into memory. If no change is desired, or after the new value has been entered, there are three options.

A. To view the contents of the next address; enter a SPACE.

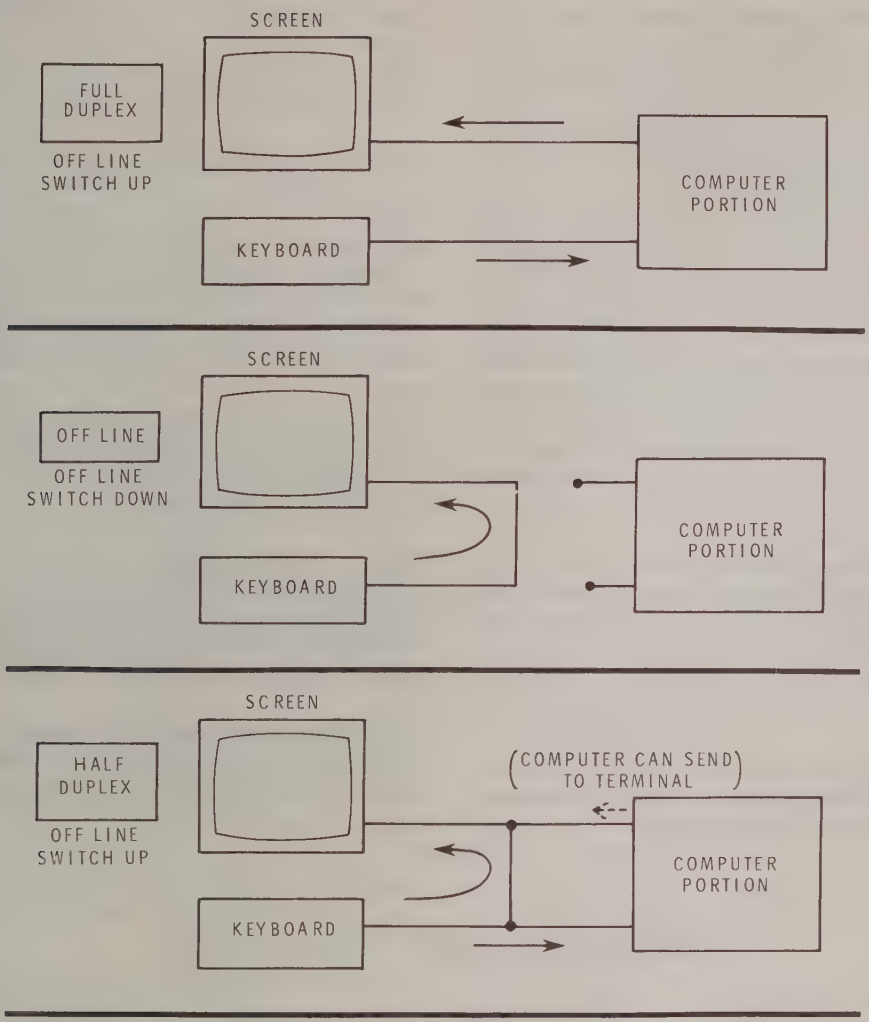
B. To view the contents of the previous address; enter a minus sign (−).

C. To exit the Substitute mode; enter a Carriage Return.

KEYBOARD OPERATION

Pictorial 6-1 (Illustration Booklet, Page 10) shows the keyboard of the Computer. The power ON/OFF switch is located on the right rear corner of the back panel. Whenever you turn on the Computer, allow the tube about 30 seconds to warm up. You should then see a flashing line (cursor) or block cursor (if it was selected) in the upper left-hand corner of the screen.

The keyboard allows you to send data to the computer or the screen. Most of the keys are the same as they are on most typewriters; they type the same alphanumeric characters. A clicking sound tells you that each keystroke has been processed. You cannot electrically damage the Computer by typing on the keys.



PICTORIAL 6-2

The screen contains 1,920 normal character positions; 24 lines of 80 characters. Only one character can occupy a character position at any given time and it will remain there until it is erased or replaced.

When the Computer is initially turned on, it clears the screen by placing spaces in all character positions. The cursor is the blinking horizontal line that appears at the home position. It underlines the character position where the next character will be written. (The block cursor will fill the character position.)

As shown in Pictorial 6-2, the Computer can be used in any one of three different modes; full duplex, off line, or half duplex. (However, half duplex is not a normal Heath mode.)

When the Computer is on line, the keyboard can transmit any one of the 128_{10} ASCII characters (see the "ASCII Characters" chart on Page 12-1) to the computer section. However, some of these characters will not be displayed if the computer sends them back to the Terminal section. (See the chart.)

In the off line mode, the keyboard is effectively disconnected from the computer and controls the screen directly. This way, you can position the cursor (↑, ↓, →, ←, and HOME), insert or delete characters or lines (IC, DC, IL, and DL), or erase (ERASE), without sending the codes to the computer — which could otherwise disrupt a program, etc.

Another way of controlling the screen without sending code to the computer is to use the CTRL key. Example: you want to erase the screen, but you do **not** want to transmit a code to the computer. Press and hold the CTRL key and then type SHIFT ERASE. This tells the terminal section to erase the screen, but not to send the code to the computer section. Again, you can use this procedure with the cursor keys (↑, ↓, →, ←, and HOME), the Insert Line, Delete Line, Insert Character, and Delete Character keys, and ERASE.

Whenever you use the special escape codes to enter and exit the special modes, make sure you enter the lower-case and upper-case letters just as they are called for in this Manual. For instance, type ESC p, not ESC P to enter the reverse video mode. See "Special Modes" on Page 6-7.

The "ASCII Characters" and "Escape Sequences" (see the "Appendix," Page 12-1) show the commands and special escape sequences that the terminal section sends and responds to. Your computer **must** contain the proper software for it to respond to and generate the codes that use these special features. Different versions of software may support different features.

The Computer has a 128 character input FIFO (first in, first out buffer) for receiving and holding characters until the terminal section can process them. In some cases (such as when the terminal section is operating at 9600 baud in the "insert character" mode), the FIFO can be filled faster than the terminal section can process the characters. In this case, the terminal section will send X OFF (control S) when the FIFO has received 112 characters. After the terminal section has processed enough characters so that only 96 characters remain in FIFO, it will send X ON (control Q) to the computer section to indicate that it is ready to accept more characters.

When the terminal section sends X OFF, this is only an indication that the buffer is nearly full. Characters will not be lost until after the FIFO has received a full 128 characters. At this point more incoming characters will be lost and the bell will sound.

Three BASIC demonstration programs are included in the "Appendix" to show you how some of the Computer features are implemented in BASIC. Enter and run them if you wish.

NORMAL MODES AND KEYS

The following descriptions are for switch S402 set to all zeros. For the operation of special functions, refer to "Special Modes and Keys" on Page 6-7.

ALPHABETIC KEYS

The Computer has the standard 26 letters of the alphabet. These keys can transmit either lower-case or upper-case codes as well as display them on the screen. You can either hold the SHIFT key down or you can push the CAPS LOCK key to obtain upper-case letters.

NONALPHABETIC KEYS

The nonalphabetic keys are those with double markings. These include the numbers 0 through 9, punctuation marks, and special characters. The lower marking is generated when both of the SHIFT keys are released, while the upper marking is generated when either (or both) SHIFT key is held down. The CAPS LOCK key will not shift these keys.

MISCELLANEOUS

RETURN — Moves the cursor to the first character position of the line that it is currently in. If the cursor is already at the first character position, it remains there. RETURN is a nondisplayable character.

LINE FEED — Moves the cursor down one line. LINE FEED is a nondisplayable character. If the cursor is at the bottom line, a LINE FEED causes it to remain there, but all of the data on the screen moves up one line. Data on the top line is lost as it is scrolled up and off the screen.

SPACE BAR — Causes the cursor to move one character position to the right. A Space is a nondisplayable character. If you type the Space Bar when the cursor is positioned below a displayed character, the character is replaced by a space and the cursor moves one character position to the right. If you type the Space Bar when the cursor is at the right end of a line, the cursor will remain there since neither a carriage return nor a line feed is generated.

BACK SPACE — Moves the cursor one space to the left. If the cursor is at the start (left end) of a line, it will not move when you type a BACK SPACE.

DELETE — Transmits the ASCII code 177. It is a nondisplayable character. Heath software uses this key to cancel the last character that was input.

TAB — Moves the cursor to the next tab stop (eight character space) to the right. The tab stops are fixed at 9, 17, 25, 33, 41, 49, 57, 65, and 73 (columns are numbered 1 through 80). If the cursor is at character position 73 through 79, it will only move one character position to the right each time you type the TAB key. If the cursor is at character position 80, it will not move when you type the TAB key (unless the wraparound feature has been selected).

ESC (Escape) — A non-displayable character that transmits the ASCII code 033. This key is used in combination with other keys to enter and exit special modes. See "Special Modes and Keys" on Page 6-7.

For a complete listing of Heath and ANSI codes using escape sequences and their definitions, refer to the Appendix (Pages 12-10 and 12-17).

REPEAT — When you hold this key in, along with another key, it will repeat the function of the other key as long as both keys are held down. The repeat rate is approximately 8-characters per second. However, if the baud rate that has been selected is less than the repeat rate, the repeat function will operate at the slower rate.

SHIFT — When you use this key in conjunction with another key, the character printed on the upper portion of that key will be displayed. When you use the SHIFT keys in conjunction with the alphabetic keys, the upper-case character is displayed.

CAPS LOCK — When this latching key is down, the terminal section will transmit the ASCII code for, and display, upper-case (capital) alphabetic letters. It does not shift the keys with the double markings.

OFF LINE — When this latching key is down, the terminal section is inhibited from transmitting or receiving data. However, any displayable characters that you type on the keyboard will appear on the screen and any control codes will be responded to.

BREAK — When you type this key, it generates a continuous space at the output of the terminal section. It is generally used to tell the computer that you wish to interrupt execution.

RESET — Allows you to reset the Computer to its preset condition; it exits all escape modes and resets the baud rate to the rate selected by the switches on the logic circuit board. To use this key, you must press only the right-hand SHIFT key and the RESET key at the same time. This two-key combination prevents you from inadvertently resetting the Terminal.

SCROLL — When this is used with Heath software, when in the Hold Screen Mode, you can type the SCROLL key to instruct the Terminal to display another line of information onto the screen. You can simultaneously press the SHIFT and SCROLL keys to display another 24 lines of information onto the screen.

CONTROL KEY

The CTRL key is held down while you push one of the other keys to send the 32 ASCII control codes to the Computer. Refer to the "ASCII Characters" chart in the "Appendix" (Page 12-1) of this Manual for a listing of the control keys. These are non-displayable characters. The Terminal responds to only seven of the control characters from the keyboard or from the serial input port. These seven characters are:

BELL (BEL or CTRL G) — Causes the Terminal to sound an audible tone through an internal speaker.

Back Space (BS or CTRL H) — Duplicates the BACK SPACE key.

Horizontal Tab (HT or CTRL I) — Duplicates the TAB key.

Line Feed (LF or CTRL J) — Duplicates the LINE FEED key.

Carriage Return (CR or CTRL M) — Duplicates the RETURN key.

Escape (ESC or CTRL [ ) — Duplicates the ESC key.

(CTRL X) — Cancels the current escape sequence.

SPECIAL MODES AND KEYS

Escape sequences allow you to use two or more keys together to provide a certain function or to get your unit to operate in a particular manner. This provides a maximum of operation with a minimum of keys.

NOTE: The following descriptions give Heath mode escape sequences. For ANSI escape sequences, refer to the "Appendix." Also, HDOS must be set to NOMLO before the example programs will run properly. See Page 12-50.

CURSOR FUNCTIONS

Cursor Home — ESC H — [Shift 5 (HOME) of keypad]
Moves the cursor to the first character position on the first line (home).

Cursor Forward — ESC C — [Shift 6 (→) of keypad]
Moves the cursor one character position to the right. If the cursor is at the end of the line, it will remain there.

Cursor Backward — ESC D — [Shift 4 (←) of keypad]
Moves the cursor one character position to the left (backspaces). If the cursor is at the start (left end) of a line, it will remain there.

Cursor Down — ESC B — [Shift 2 (↓) of keypad]
Moves the cursor down one line. If the cursor is at the bottom line, it will remain there; however, a scroll will not occur.

Cursor Up — ESC A — [Shift 8 (↑) of keypad]
Moves the cursor up one line. If the cursor is at the top line, it will remain there; however, a scroll will not occur.

Reverse Index — ESC I — This is a reverse line feed. It causes the cursor to move upward one line. If the cursor is at the top line it will remain there. However, any text on the screen will be scrolled downward one line.

Cursor Position Report — ESC n — Reports the position of the cursor in the form of ESC Y line# column#. The following BASIC program gives an example of its use.

```
00010 PRINT "PRESS RETURN";CHR$(27);"n"
00020 LINE INPUT ;A$
00030 B$=LEFT$(A$,1)
00040 A$=RIGHT$(A$,LEN(A$)-1)
00050 PRINT ASC(B$),
00060 IF LEN(A$)>0 THEN 30
00070 END
```

When you run the program and push the RETURN key, the computer will respond with the following decimal numbers:

27 89 55 44

Here the 27 equals ESC, 89 equals Y, 55 is the line# (55−31=24), and 44 is the column# (44−31=13). (See "Direct Cursor Addressing" below.) Therefore, the reported cursor position is:

ESC Y line# 24 column# 13

Save Cursor Position — ESC j — The present cursor position is saved so the cursor can be returned there later on the "Set to previously saved position" command. "Demonstration Program #2" in the "Appendix" of this Manual gives an example of this feature in a BASIC program.

Set To Previously Saved Position — ESC k — Returns the cursor to the position where it was when it received the last "Save cursor position" command.

Direct Cursor Addressing — ESC Y — Allows the computer to control the position of the cursor on the screen by entering the escape code, the ASCII character which represents the line number, and the ASCII character which represents the column number.

The first line and the left column are both 32_{10} and increase from there. The number 32_{10} is used because it is the smallest value of the printing characters. All values less than 32_{10} are control codes, which can interfere with operating sequences of some computers.

Since the lines are numbered from 1 to 24 (from top to bottom) and the columns from 1 to 80 (from left to right), you must add the proper line and column numbers to 31_{10}. Then convert these decimal numbers to their equivalent ASCII characters and enter them in the following order:

ESC Y line# (ASCII character) column# (ASCII character)

For example, to place the cursor at line 20, column 40, you will first have to add 31_{10} to the line number to find the value of the line#.

$31 + 20 = 51_{10}$

Then use the "ASCII Characters" chart (in the "Appendix") to find the ASCII character that corresponds to 51_{10}. In this case, it is the number 3. Next, add 31_{10} to the column number to find the actual value of the column #.

$31 + 40 = 71_{10}$

Again, use the ASCII chart to find the ASCII character that corresponds to 71_{10}, which is the symbol G.

To demonstrate this example, make sure the OFF LINE key is down. Then type ESC Y 3 G. The cursor should move to line 20, column 40.

If you specify a line# that does not exist on the screen, the cursor will remain in the line it is presently in. If you specify a column# that does not exist on the screen, the cursor will move to the right-most column.

"Demonstration Program #1" in the "Appendix" of this Manual shows you how this feature is used in a BASIC program.

ERASING AND EDITING

Clear Display (SHIFT ERASE) — ESC E — Erases all the information on the screen. The screen is filled with spaces and the cursor is placed in the home position.

"Demonstration Program #1" in the "Appendix" of this Manual shows you how this feature is used in a BASIC program.

Erase Beginning Of Display — ESC b — Erases the display from the start of the screen to the cursor position, and includes the cursor position.

Erase To End Of Page (ERASE Key) — ESC J — Erases all the information from the cursor (including the cursor position) to the end of the page.

Erase Entire Line — ESC l — Erases the entire line, including the cursor position.

Erase Beginning Of Line — ESC o — Erases from the beginning of the line to the cursor position, and includes the cursor position.

Erase To End Of Line — ESC K — Erases from the cursor (including the cursor position) to the end of the line.

Insert Line — ESC L — [Shift 1 (IL) of keypad]
Inserts a new blank line by moving the line that the cursor is on, and all following lines, down one line. Then the cursor is moved to the beginning of the blank line.

Delete Line — ESC M — [Shift 3 (DL) of keypad]
Deletes the contents of the line that the cursor is on, places the cursor at the beginning of the line, moves all the following lines up one line, and adds a blank line at line 24.

Delete character — ESC N — [Shift 9 (DC) of keypad]
Deletes the character at the cursor position and shifts any existing text that is to the right of the cursor, and on the same line, one character position to the left.

Enter Insert Character Mode — ESC @ — [shift 7 (IC) of keypad]
Lets you insert characters or words into text already displayed on the screen. The first time you type IC, the Terminal enters the Insert Character Mode. You can then use the cursor controls to place the cursor at the point where you want to insert characters. As you type in the desired characters, any existing text directly above and to the right of the cursor is shifted to the right. This feature lets you add letters or words to existing text without having to re-type the whole text. When you finish inserting characters, type IC again to exit the Insert Character Mode. The Terminal transmits an ESC @ to enter, and an ESC O to exit the Insert Character Mode.

Exit Insert Character Mode — ESC O — Exits the Insert Character Mode. See "Enter Insert Character Mode" above.

CONFIGURATION

Reset To Power-Up Configuration — ESC z — Nullifies all previously set escape modes and returns to the power-up configuration set by switches S401 and S402 on the terminal logic circuit board.

Modify The Baud Rate — ESC r — Initially, the baud rate is set by the switches on the terminal logic circuit board. However, you can change the baud rate from the keyboard. To do this, type ESC r followed by the appropriate letter given below:

A=110	G=2000
B=150	H=2400
C=300	I=3600
D=600	J=4800
E=1200	K=7200
F=1800	L=9600

Example: If the baud rate switches on the terminal logic circuit board are set to 4800 baud and you want to communicate with the computer at 9600 baud, just type ESC r L.

The baud rate reverts back to the baud rate set by the switches on the circuit board when you RESET the Terminal (RESET and right-hand SHIFT keys) or when you turn the Terminal off and then back on.

Set Mode — ESC x — Certain operating modes can be enabled and disabled from the keyboard. To enable the functions, type ESC x followed by the appropriate number given below:

1 = Enable 25th line. The 25th line is available as a line that is totally separate from the normally-used 24 lines. You might use this line, for example, to identify the user function keys with labels which correspond to the function that your computer provides when it receives these function key escape codes. Or you might use it to display information concerning the status of your computer while a program is running.

The only way to place the cursor on the 25th line is to enable the 25th line and then use "Cursor Addressing." Once on the 25th line, the terminal acts like a 1-line terminal ("erase in display" commands only operate on the 25th line) until you use cursor addressing to place the cursor on one of the other 24 lines of the Terminal. This is a good place to use the "Save Cursor Position" and the "Set Cursor To Previously Saved Position" routines. With these routines, the current cursor position can be saved, your routine can address the 25th line, write information on the 25th line, and return to the "remembered" cursor location without your program having to remember that location. "Demonstration Program #2" in the "Appendix" of this Manual gives an example of these features in a BASIC program.

Also, when the cursor is on the 25th line, all erase functions affect only this line and a line feed will not cause a scroll.

2 = No key click. This function turns off the key click.

3 = Hold screen mode. See "Enter Hold Screen Mode" (Page 6-11) for a description of this function.

4 = Block cursor. Produces a cursor that fills the entire character position.

5 = Cursor off. Turns off the cursor so there is no cursor at all.

6 = Keypad shifted. See "Enter Keypad Shifted Mode" for a description of this function.

7 = Alternate keypad mode. See "Enter Alternate Keypad Mode" for a description of this function.

8 = Auto line feed on receipt of CR. A line feed is automatically performed (in addition to a CARRIAGE RETURN) when a CARRIAGE RETURN is received.

9 = Auto CR on receipt of line feed. A CARRIAGE RETURN is automatically performed (in addition to a line feed) when a line feed is received.

Example: If you want to turn off the cursor, type ESC x 5.

These functions default back to their initial states (as set by switches S401 and S402 on the terminal logic circuit board) when the Terminal is reset (RESET and right-hand SHIFT keys) or when you turn the Terminal off and then back on again. You can also reset these functions using the Reset Mode escape codes (ESC y). See below.

Reset Mode — ESC y — Resets the "Set Mode" functions to their power-up default states. To reset a function, type ESC y followed by the appropriate number given below.

1 = Disable 25th line
2 = Enable key click
3 = Exit hold screen mode
4 = Underscore cursor
5 = Cursor on
6 = Keypad unshifted
7 = Exit alternate keypad mode
8 = No auto line feed
9 = No auto CR

See "Set Modes" above.

Enter ANSI Mode — ESC < — Enters the ANSI mode. See the "Appendix" in the rear of this Manual for the definition and descriptions of the ANSI mode escape codes.

MODES OF OPERATION

Enter Hold Screen Mode — ESC [— The Hold Screen Mode allows you to control when new information is printed on the screen. This is especially useful when you are reading lists or looking for a particular part of a program. Push the OFF LINE key to its down position and then type ESC [to enter the Hold Screen Mode. Then after you release the OFF LINE key, each time you type the SCROLL key, a new line of text will appear on the bottom line and the top line of text will scroll up and off the screen. If you type SHIFT SCROLL, a whole new page (24 lines) of text will be scrolled onto the screen. Press the OFF LINE key to its down position and type ESC \ to exit the Hold Screen Mode. Remember, that when the cursor is at the start of a line of text, the Terminal is probably waiting for a scroll command when in this mode.

This mode requires that the host computer respond to XON and XOFF.

Exit Hold Screen Mode — ESC\ — Exits the Hold Screen Mode. See "Enter Hold Screen Mode" above.

Enter Reverse Video Mode — ESC p — The characters displayed on the screen can also be displayed in reverse video, a black character on a white background. Type ESC p to enter the Reverse Video Mode, and ESC q to exit the Reverse Video Mode.

The following BASIC program shows you how to send the escape codes to the Terminal to enter and exit the reverse video mode.

```
00010 REM Reverse Video Demonstration
00020 PRINT "This is a demonstration of the ";
00030 PRINT CHR$(27);"P";
00040 PRINT "reverse video";
00050 PRINT CHR$(27);"q";
00060 PRINT " feature."
00070 END
```

Exit Reverse Video Mode — ESC q — Exits the Reverse Video Mode. See "Enter Reverse Video Mode" above.

Enter Graphics Mode — ESC F — The graphics mode lets you display 33 special symbols. Refer to the "Graphic Mode Symbols" in the "Appendix" of this Manual. Type ESC F to enter the Graphics Mode. Then type any of the 26 lower-case keys or the seven other symbol keys that correspond to the graphic symbols. Type ESC G to exit the Graphics Mode. You can place the Terminal in the Reverse Video Mode while it is in the Graphics Mode to increase the number of graphic symbols.

"Demonstration Program #1" in the "Appendix" of this Manual shows you how this feature is used in a BASIC program.

Exit Graphics Mode — ESC G — Exits the Graphics Mode. See "Enter Graphics Mode" above.

Enter Keypad Shifted Mode — ESC t — The shifted functions that the keypad transmits normally require you to press and hold the SHIFT key when you type one of the keys. You can type ESC t to enter the Shifted Keypad Mode so that you do not need to hold the SHIFT key to obtain the unshifted functions. However, if you place the Terminal in the Shifted Keypad Mode and you need to use the unshifted functions (numbers), you will have to press and hold the SHIFT key to obtain them. Type ESC u to exit the Shifted Keypad Mode.

Exit Keypad Shifted Mode — ESC u — Exits the Keypad Shifted Mode. See "Enter Keypad Shifted Mode" above.

Enter Alternate Keypad Mode — ESC = — The codes sent to the computer from the Terminal Keypad normally include the numbers, period, ENTER, and (when shifted) some special cursor movement and editing functions. You can change these keypad codes using the Alternate Keypad Mode to transmit specific escape codes that your computer may respond to.

Type ESC = to enter and ESC > to exit the Alternate Keypad Mode.

The following chart lists the escape codes sent by the Terminal in the Alternate Keypad Mode.

KEY	HEATH ESCAPE CODE	ANSI ESCAPE CODE
0	ESC ? p	ESC O p
1	ESC ? q	ESC O q
2	ESC ? r	ESC O r
3	ESC ? s	ESC O s
4	ESC ? t	ESC O t
5	ESC ? u	ESC O u
6	ESC ? v	ESC O v
7	ESC ? w	ESC O w
8	ESC ? x	ESC O x
9	ESC ? y	ESC O y
•	ESC ? n	ESC O n
ENTER	ESC ? M	ESC O M

Exit Alternate Keypad Mode — ESC > — Exits the Alternate Keypad Mode. See "Enter Alternate Keypad Mode" above.

ADDITIONAL FUNCTIONS

Keyboard Disabled — ESC { — Inhibits the output of the keyboard. WARNING: After this function is entered, the keyboard can only be turned back on by your program or a master reset.

Keyboard Enabled — ESC { — A computer-sent code that enables the keyboard after it was inhibited by a "Keyboard Disabled" command.

Wrap Around At End Of Line — ESC v — 81st character on a line is automatically placed in the first character position on the next line. The page scrolls up if necessary.

Discard At End Of Line — ESC w — After the 80th character in a line, the characters overprint. Therefore, only the last character received will be displayed in position 80.

Identify As VT52* (ESC / K) — ESC Z — The Terminal responds to interrogation with ESC / K to indicate that it can perform as a VT52.

Transmit 25th Line — ESC] See "Transmit Page" below.

Transmit Page — ESC #
The transmit functions (Transmit 25th Line and Transmit Page) are the same except for the source of the data transmitted. Your computer may have a special routine (which is required for this function to work) so it can accept the transmitted codes.

*Registered Trademark, DEC Corporation

Basically (assuming that the mode has not changed), the data is transmitted the same as it appears on the CRT. This includes all 1920 characters (24 lines of 80 characters), or the 80 characters of the 25th line. However, it is possible that the actual number of characters transmitted will be more than 1920. If graphic characters, reverse video characters, or both are encountered, the proper escape sequence for entering the respective modes will be transmitted. When one or both of these parameters no longer apply, the appropriate escape sequence will then be sent to exit the mode.

The escape sequence which is sent is determined by whether the Terminal is in the Heath mode or the ANSI mode. The sequence will be the same as that which was sent to the Terminal (or entered from the keyboard) to cause the Terminal to enter and/or exit the reverse video and graphic character modes.

Other than the above required escape sequences, the only characters sent are the displayed characters.

Following the transmission of the last character, a CARRIAGE RETURN is sent and the bell will sound.

If a transmit page is executed (ESC #), only lines 1 through 24 are transmitted. If you want to transmit the 25th line, you must ask for that specifically (ESC]). This operates the same as the transmit page except that only the 80 characters of the 25th line (and any necessary escape sequences) are transmitted and followed by a CARRIAGE RETURN. In the event that the 25th line is not enabled, only a CARRIAGE RETURN will be transmitted.

Special Function Keys

The eight special function keys (f1, f2, f3, f4, f5, Blue, Red, and White) on the top row of the keyboard transmit two-character escape codes to the computer. You can define the meanings of each of these keys to suit your particular application (your software program must recognize the particular escape codes associated with the keys). See the "Appendix."

SUMMARY OF KEYPAD FUNCTIONS

The keypad can operate in any one of four modes: normal unshifted, normal shifted, alternate unshifted, and alternate shifted. Then, within each of these modes, you can use the SHIFT key shifted or unshifted as a toggle. (See "Enter Keypad Shifted Mode" and "Enter Alternate Keypad Mode.")

Normal Unshifted — This is the normal operating mode.

Example:	TYPE	TERMINAL TRANSMITS
	3	3
	SHIFT 3	DL (Delete Line)

Normal Shifted — ESC t to enter; ESC u to exit — The normal functions are inverted.

Example:	TYPE	TERMINAL TRANSMITS
	3	DL (Delete Line)
	SHIFT 3	3

Alternate Unshifted — ESC = to enter; ESC > to exit — This is the normal alternate mode.

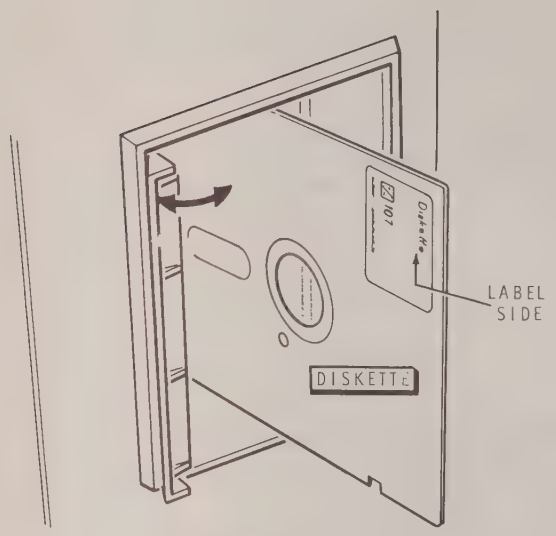
Example:	TYPE	TERMINAL TRANSMITS
	3	ESC ? s (Heath escape code)
	SHIFT 3	DL (Delete Line)

Alternate Shifted — ESC t ESC = to enter; ESC u ESC > to exit — The normal alternate functions ae now inverted.

Example:	TYPE	TERMINAL TRANSMITS
	3	DL (Delete Line)
	SHIFT 3	ESC ? s (Heath escape code)

See the "Appendix" for actual codes sent and for ANSI codes.

DISK OPERATION



PICTORIAL 6-3

DISKETTE LOADING

Refer to Pictorial 6-3, open the front panel door, and insert the diskette, label towards the screen. Then close the door.

DISKETTE HANDLING

The diskette can be easily damaged. Handle it carefully as follows:

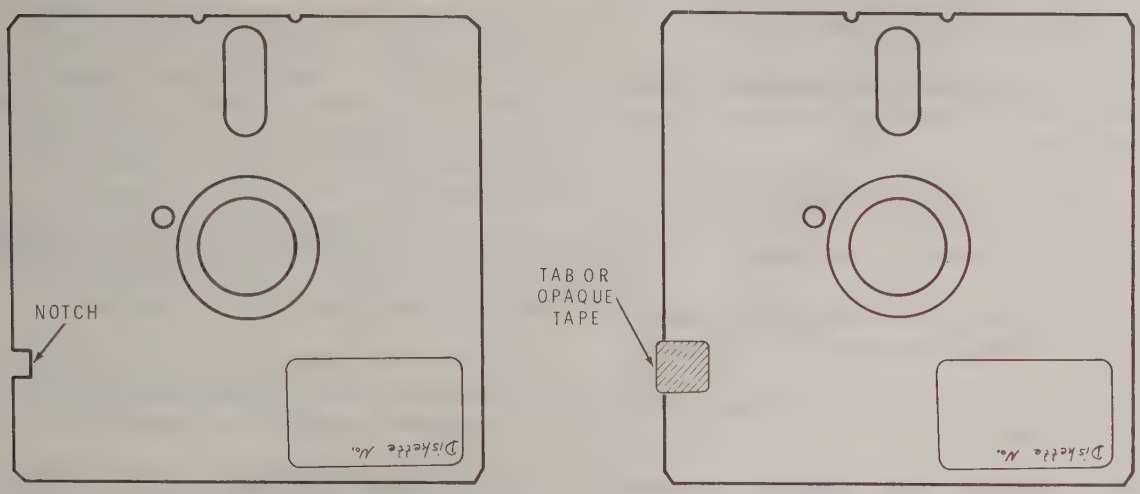
1. Keep the diskette in its storage envelope whenever it is not in the Floppy Disk drive.
2. Keep the diskette away from magnetic fields. Magnetic fields can distort the recorded data on the diskette.
3. Replace damaged or worn storage envelopes.
4. Write on the plastic jacket only with a felt-tip pen. Do not use a lead pencil or ball-point pen.
5. Keep the diskette away from hot or contaminating materials.
6. Do not expose the diskette to sunlight.
7. Do not touch or clean the surface of the diskette. Abrasions can alter stored data.

WRITE-PROTECT

This diskette can be write protected so that it cannot be written on. To do this, cover the side notch with a tab or opaque tape. See Pictorial 6-4.

SPEED TEST

See "Rotational Speed Test" on Page 4-9.



PICTORIAL 6-4

CASSETTE OPERATION

COPYING TAPES

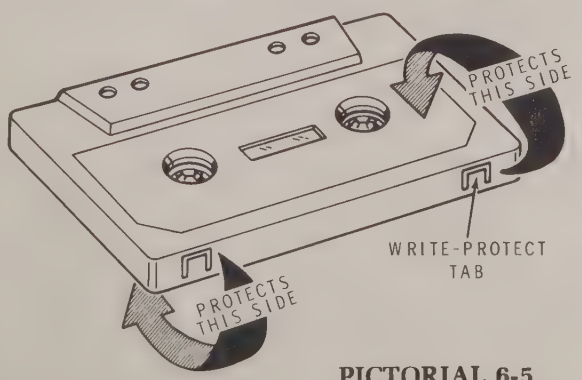
The Cassette Interface provides the greatest reliability when tapes are played back by the same machine on which they were recorded. Therefore, it is recommended that you make a copy of the H8 software tape. This will give you maximum reliability when loading tapes and the original can be used as a backup copy if something happens to the work copy. See Pictorial 6-5 (Illustration Booklet, Page 10).

Data is stored on magnetic tape one bit at a time as the tape moves past the record head. The data rate is 1200 bits per second. Since the tape moves at 1-7/8" per second, at 1200 bits per second, there will be 640 bits on each inch of tape. This means that each bit is only .0016 inches long. If a single bit is lost, the program will not run. Obviously, small defects in the tape which are not noticeable on audio recordings will be disastrous to a data storage tape. Tape defects, dust, smoke, and improper tape handling will all cause loss of data. Therefore, be very careful with your tapes.

To keep from accidentally erasing a tape, use a screwdriver or other suitable tool and break out the write-protect tab. See Pictorial 6-5.

Choose a good recorder. One of the two biggest factors in reliability is the recorder you select. It will need an external speaker or earphone output jack and an auxiliary, radio, or microphone input jack. A motor control jack will make your system easier to operate but it is not essential. Preferably, choose a recorder that is designed to record music. Clean the tape head regularly and be sure it is free of dust.

Tape is the other biggest factor in reliability. Choose top quality tape in a well-designed housing. Use the type of tape your recorder was designed for, chromium dioxide if possible. Choose one of the shorter tapes (C-30); the tape is thicker.



PICTORIAL 6-5

To protect your tapes, follow these precautions:

1. Always rewind a tape before you remove it from the recorder. This prevents contamination and scratching.
2. Always store a tape in its protective case.
3. Do not touch the tape or the leader.
4. Never place tapes near magnetic fields, as generated by motors, magnets, or TV SETS.
5. Do not expose tapes to direct sunlight or high temperatures.
6. Always have duplicate copies of all valuable tapes; like system software tapes.

All Heath computer hardware and software products were designed to work together as a complete system. Proper operation can be assured only when the computers are used with Heath designed or approved accessories. Heath does not assume the responsibility for improper operation resulting from custom interfacing, custom software, or the use of accessories not approved by Heath Company.

When you set up your system, you may have to experiment to find the best settings for the volume control, tone control, or both. The tone control is normally set to midrange or toward the treble end. The volume control is not critical on playback once the minimum level is reached. Increasing the volume will sometimes be helpful when using poor recordings.

If your recorder has an automatic level control, defeat it if possible. Use the record level control to set the level to the highest level before overloading.

If you want to use a cassette deck that does not have a power amplifier, remove programming jumper JP705 from the cassette interface circuit board.

If you have two tape machines, you can use one for playback while you record on the other. To do this, use the audio cable and the 2-wire cable with black tubing for the playback machine. Use the other two cables for the recording machines. See the inset drawing on Pictorial 6-5.

USE AS A TERMINAL

To use your Computer as a Terminal, refer to Pictorial 6-6 (Illustration Booklet, Page 11) and:

( ) Disconnect the cable from P404 on the back of the terminal logic circuit board.

( ) Disconnect cable #134-1070 from plug P605 (the bottom one) on the serial interface circuit board.

( ) Plug the cable into P605 on the back of the terminal logic circuit board (brown wire up).

READJUSTMENT

This section contains several adjustments that you may need to make to properly maintain your Computer. You will have to remove or tilt back the cabinet shell in order to reach the controls, coils, and adjustments called for in this section. To do this, refer to the inset drawing on Pictorial 3-1 and carefully remove the cabinet shell back.

( ) On the terminal logic circuit board (see Pictorial 5-1, Illustration Booklet, Page 8), set section 2 of switch S402 down to its 1 position.

NOTE: When power is turned on, do not touch the flyback transformer, the high voltage lead, or the anode socket at the back of the CRT, as it is possible to receive an electrical shock from these areas. Also, to lessen the chances of an electrical shock while you are making adjustments, keep your other hand away from this unit and all other metallic objects.

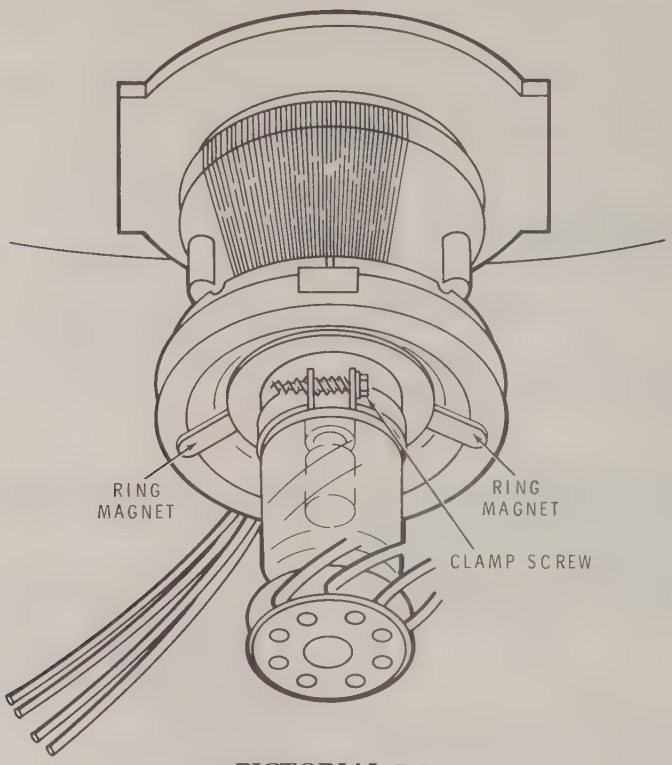
( ) Plug in the line cord and set the POWER switch to ON.

Refer to Pictorial 7-1 (Illustration Booklet, Page 11) for the locations of controls on the video circuit board.

( ) After a short warm-up time, a light raster should appear on the screen. If it does not, adjust G1 control R268 counterclockwise (as viewed from the left side), to cause the raster to appear.

( ) If the display is slanted, loosen the yoke clamp screw slightly and slowly turn the yoke to properly line up the raster on the screen. See Pictorial 7-2 on Page 7-2.

( ) Adjust the VERT SIZE control R226 (on the video circuit board) so the raster is approximately 6″ high.



PICTORIAL 7-2

( ) Refer to Pictorial 7-2 and rotate the ring magnets on the back of the yoke to center the display on the screen.

( ) Adjust BRIGHTNESS control R216 until a blinking cursor (underline) appears at the top corner of the screen.

( ) Set the OFF LINE and CAPS LOCK keys to their down positions.

( ) Hold the "Z" key and the REPEAT key down and fill the screen with characters.

( ) Adjust HORIZ CENTERING control R253 to center the display horizontally within the raster.

( ) Adjust VERT LINEARITY control R228 so that the top and bottom rows of characters are of uniform size.

NOTE: You should make the next adjustment in a darkened room.

( ) Turn G1 control R268 clockwise (as viewed from the left) until the raster just disappears.

( ) If the display width is not approximately 8-1/2", adjust WIDTH coil L204 to correct the width size.

( ) Adjust BRIGHTNESS control R216 to obtain the brightness that is most suitable to you.

( ) Adjust FOCUS control R271 for the best focus.

( ) Recheck the display for proper alignment of the screen. If necessary, rotate the yoke a small amount. Then tighten the yoke clamp screw only enough to hold the yoke from turning.

( ) Set the POWER switch to OFF and disconnect the line cord.

( ) Set section 2 of switch S402 (on the terminal logic circuit board) up to its 0 position.

( ) Resecure all parts in your Computer and replace the cabinet shell.

TROUBLESHOOTING

IMPORTANT NOTICE

All Heath computer hardware and software products were designed to work together as a complete system. Proper operation can be assured only when the computers are used with Heath designed or approved accessories. Heath does not assume the responsibility for improper operation resulting from custom interfacing, custom software, or the use of accessories not approved by Heath Company.

Factory wired circuit boards have been wired and tested at Heath Company. If one malfunctions during the 90-day warranty period, return the complete circuit board assembly to Heath Company or a Heathkit Electronic Center. It will be promptly repaired and returned. Individual replacement parts are not supplied under warranty. **DO NOT** attempt to service this circuit board assembly yourself during the warranty period; to do so voids the warranty.

For out-of-warranty circuit boards, you can have them repaired by Heath Company (or a Heathkit Electronic Center), or you can purchase individual replacement parts to do your own service.

This section of the Manual is divided into two parts. The first part titled "General Troubleshooting Information," pertains to your Computer only if you built it from a kit. It describes what to do about difficulties that may occur during or right after you assemble the Computer.

The second part, titled "Troubleshooting Charts," lists problems or conditions that might occur. The "Possible Cause" column lists the components associated with the problem. This will help you relate a problem to the Schematic and Circuit Description.

Refer to the "Circuit Board X-Ray Views" (Illustration Booklet, Pages 16 thru 21) for the physical location of parts on the circuit boards.

Due to the complexity of the factory wired circuit boards and the closed loop configuration of the control and interface circuits, we recommend that you return these parts for repair if necessary. Refer to the "Customer Service" information inside the rear cover of this Manual.

GENERAL TROUBLESHOOTING INFORMATION

This section of the Manual applies to your Computer only if you assembled it from a kit.

NOTE: The following checks will be most effective if you apply them to one part of the kit at a time.

1. Recheck the wiring. Trace each lead in colored pencil on the Pictorial as it is checked. It is frequently helpful to have a friend check your work. Someone who is not familiar with the unit may notice something you have consistently overlooked.

2. About 90% of the kits that are returned for repair do not function properly due to poor connections and soldering. Therefore, you can eliminate many troubles by carefully inspecting the connections to make sure they are soldered as described in the "Soldering" information at the beginning of the Assembly Manual. Reheat any doubtful connections. Be sure all wires are soldered at places where several wires are connected.

3. Check each circuit board foil to be sure there are no solder bridges between adjacent connections. Remove any solder bridges by holding a clean soldering iron tip between the two points that are bridged until the excess solder flows **down** onto the tip of the soldering iron.

4. Check each resistor value carefully. A resistor that is discolored, or cracked, or shows any sign of bulging would indicate that it is faulty and should be replaced.

5. Be sure each diode is carefully installed with the banded end positioned correctly.

6. Check all component leads connected to the circuit boards. Make sure the leads do not extend through the circuit board and come in contact with other connections or parts.

7. The components listed in the "Possible Cause" column of the "Troubleshooting Chart" are the most likely causes (but not necessarily the only cause) of a problem. When you check these components, look first for the following items:

   — Parts installed incorrectly or backwards. This pertains especially to diodes, electrolytic and tantalum capacitors, and integrated circuits.

   — Unsoldered or inadequately soldered parts. Reheat the connections in the area of a problem.

   — Incorrect or interchanged parts. Check the part numbers on the diodes and integrated circuits.

8. Check the set-up switches on the logic circuit boards to be sure that they are in their proper positions. See Pictorials 5-1 and 5-2 (Illustration Booklet, Pages 8 and 9).

NOTE: In an extreme case where you are unable to resolve a difficulty, refer to the "Customer Service" information inside the rear cover of the Manual. Your "Warranty" is located inside the front cover.

TROUBLESHOOTING CHARTS

The following charts list conditions and possible causes of several specific malfunctions. If a particular part is mentioned (Q213 for example) as a possible cause, check that part and other components connected to that part to see that they are installed and/or wired correctly. Also check for solder bridges and poor connections in the surrounding area. It is also possible, on rare occasions, for a part to be faulty and require replacement.

WARNING: Measure the anode voltage only with an approved high voltage probe.

CAUTION: Never operate the Computer unless the short black ground wire coming from the corner of the video board is connected to the CRT ground.

POWER SUPPLY PROBLEMS

CONDITION	POSSIBLE CAUSE
Nothing happens at turn on.	1. Not plugged in. 2. Fuse F1 blown. 3. Primary/line cord wiring. 4. Switch SW1 wiring. 5. Fuseholder wiring. 6. Power transformer T1.
Fuse blows.	1. Check primary wiring. 2. Short circuit on power supply circuit board. 3. Short circuit across transformer secondary. 4. Diodes D101-D112, and BR1. 5. C1, C102-C104. 6. U401-U405. 7. Short between collector of Q213 and video board heat sink. 8. Incorrect fuse. 9. Power transformer T1.
No output from 5 V supplies, or voltage(s) too high or too low.	1. U401, U402. 2. D105-D107. 3. C103.
No +12 V, or is too high or too low.	1. U403. 2. D101-D104. 3. C102.
No −12 V, or is too high or too low.	1. U404. 2. D101-D104. 3. C104.
No −5 V, or is too high or too low.	1. U405. 2. U404 (−12 V source supplies −5 V regulator). 3. D101-D104. 4. C104.
No +53 V, or is too high or too low.	1. Q213, Q211, Q209. 2. D203, D204.
No unregulated voltages (+65, +8.5, +/−16) on power supply board.	1. Check appropriate secondary of T1, diode bridges or filter capacitor.
No anode voltage when other voltages are OK.	1. No sync pulses coming from terminal logic board. 2. Q217, Q216. 3. Deflection yoke, L204, L205. 4. C229, C233. 5. D208. 6. U201, U202.
+500 V supply is too high or too low.	1. D211. 2. C232.
−90 V supply is too high or too low.	1. D209. 2. R266. 3. C231. 4. Q218.
−6 V supply is too high or too low.	1. D205. 2. R212.

VIDEO RELATED PROBLEMS

CONDITION	POSSIBLE CAUSE
No video (blank screen).	1. Brightness control (R216) turned down. 2. Anode voltage incorrect. 3. Grid voltages incorrect (G1, G2, G4). 4. No cathode drive. 5. Q214, Q215. 6. No video signal coming from terminal logic board. 7. U479. 8. Video circuits on logic board. 9. D206. 10. No sync pulses coming from logic board. 11. D212.
Screen all white (raster).	1. Grid voltages. 2. Q214, Q215. 3. Video circuits on terminal logic board. 4. Anode voltage incorrect. 5. Q218.
Insufficient brightness.	1. Q214, Q215. 2. D205. 3. C205, C206. 4. R216 (brightness control), R214, R218, R219, R217. 5. Grid voltages.
One bright horizontal line on screen.	1. Vertical amplifier (Q203-Q208). 2. D202. 3. Deflection yoke (vertical). 4. Vertical sweep generator (Q201, Q202). 5. D201. 6. No vertical sync pulses coming from logic board. 7. U479.
Too much or too little height.	1. Vertical amplifier or sweep generator. 2. C215. 3. R248. 4. R226 (vertical size) misadjusted, open, or shorted. 5. C213.
Too much or too little width.	1. Adjust width coil L204. 2. C229, C233. 3. Deflection yoke (horizontal), L204, L205. 4. +53 V supply not correct. 5. Flyback transformer T2. 6. Q217.
Picture tube filament does not glow.	1. No horizontal sync pulses coming from logic board. 2. Filament winding of T2 (brown wires). 3. R264.
Horizontal centering does not work.	1. U201. 2. R253. 3. C221.
No horizontal sweep, but sync pulses are present at P202-1.	1. D207. 2. U201, U202. 3. Q216, Q217. 4. T201. 5. +6 V supply not correct. 6. C227. 7. D208. 8. +53 supply not correct. 9. Deflection yoke, L204, L205.

SHIPPING INFORMATION

In an extreme case where you are unable to resolve a difficulty, you may want to take your Computer to your local Heathkit Service Center or ship it to the Heath Company.

If you can isolate the problem to a particular circuit board, take (or send) only that circuit board for repair. This will save shipping cost and service expense.

Whenever possible, take your Computer (or circuit board) to your local Heathkit Electronic Center for service. Consult your Heathkit Catalog for the locations of the Electronic Centers.

However, if it becomes necessary to ship the complete unit to the Heath Company, refer to the "Customer Service" information inside the rear cover of the Manual and securely pack the Computer. Be sure to close the cabinet shell and latch it securely.

IMPORTANT: Include the following information with your Computer. It will be helpful in diagnosing and repairing your unit.

A. The problem you are having.

B. Name and model of your computer system.

C. Baud rate.

D. System configuration.

E. Any additional information that will help describe your system.

CIRCUIT DESCRIPTION

Refer to the Schematic Diagram (fold-in) and the Terminal Block Diagram (Illustration Booklet, Page 14) while you read this "Circuit Description."

To help you locate parts in the Computer or on the Schematic, the circuit component numbers (R1, C101, L301, etc.) for resistors, capacitors, coils, transistors, and integrated circuits are in the following groups.

0-99 Parts mounted on the molded chassis base or bezel.

100-199 Parts mounted on the power supply circuit board.

200-299 Parts mounted on the video circuit board.

300-399 Parts mounted on the keyboard circuit board.

400-499 Parts mounted on the terminal logic circuit board.

500-599 Parts mounted on the CPU logic circuit board.

600-699 Parts mounted on the serial interface circuit board.

700-799 Parts mounted on the cassette interface circuit board.

800-899 Parts mounted on the floppy disk interface circuit board.

POWER SUPPLY CIRCUIT BOARD

The primary circuit of the power supply consists of slow-blow fuse F1, ON/OFF switch SW3, 120 V/240 V switch SW1, NOR/LOW line switch SW2, and the primary windings of transformer T1.

The red secondary windings of transformer T1 supply AC to diode bridge rectifier D109-D112. The 65-volt rectified output of the bridge is filtered by capacitor C1. It is used to power the video circuits.

The yellow secondary winding of T1 supplies AC to the diode bridge rectifier BR1. The rectified output of the bridge (approximately 9 VDC) is filtered by capacitors C101 and C103 and is used on the logic circuit boards.

The green secondary windings supply center-tapped 30 VAC to diode bridge rectifier D101-D104. The rectified outputs of the bridge (±16 VDC) are filtered by capacitors C102 and C104. These outputs are used on the logic circuit board.

INTERCONNECTION AND GROUNDING

The three power supplies (+65, +9, and ±16) are not interconnected on the power supply circuit board. Instead, they pick up their appropriate grounds at the circuit boards they power. The +65-volt video supply connects to + and ground points on the video circuit board. The external conductive coating of the CRT and the CRT socket arc-ring both connect directly to the video circuit board ground.

The +9-volt and ±16-volt supplies connect directly to the logic circuit boards with no common grounds until they meet at the terminal logic circuit boards.

This grounding method produces two independent operating systems that do not interact with each other except through the signal ground and sync/video inputs. In the event of a CRT arc, the arc discharge current is confined to the video circuit board and it does not induce transients into the logic circuits.

The logic/video system is also floating with respect to the ground wire of the power cord. The protective ground input (pin 1) of the EIA RS232 connector connects to the power cord ground, along with all the exposed metal surfaces.

The signal ground input (pin 7) of the EIA RS232 connector connects to the logic circuit board ground.

VIDEO CIRCUIT BOARD

POWER SUPPLY

The unregulated 65-volts DC from the power supply circuit board enters the video circuit board at plug P202, pins 2 and 3. Assume that the Computer has just been turned on and the output of the +53-volt regulator is at zero volts. The base current of Q209 is supplied through resistors R201, R202, and R203. The collector current of Q209 causes Q211 to turn on and supply current to the base of Q213. As the output voltage at the emitter of Q213 rises, D204 begins to supply current to zener diode D203 through resistor R202. D203 stabilizes at 12.8 volts and provides a reference for the output voltage. The divider formed by resistors R207 and R208 samples the output voltage as it continues to rise, and applies a fraction of the voltage to the emitter of Q209. When the emitter voltage of Q209 reaches 12.15 volts (12.8-.65), its collector current is reduced to a value that keeps the output voltage stabilized at +53 volts.

Resistor R211 samples the current supplied to the load by Q213, and generates a voltage that is applied to the base of Q212. If the current exceeds about 1.1 amperes, Q212 turns on and shunts current from the base of Q211, which, in turn, prevents the output current from exceeding 1.1 amperes.

D205 and R212 form another zener regulator that supplies 6.2 volts DC to the video amplifier and 6 volts DC to the horizontal section through R213.

VIDEO AMPLIFIER

The video amplifier is a conventional cascode amplifier. The video signal enters the circuit board at plug P202, pin 4. The level of this video signal is set by Brightness control R216. It is then applied through R217 to the base of Q215. The base of Q214 is biased by the 6.2-volt supply. The signal at the collector of Q214 is coupled through resistor R1 to the cathode of the CRT. L201, which is in series with collector load resistor R215 provides high frequency compensation.

D214 and C241 make certain that the cathode voltage stays at 53 volts and keeps the CRT beam turned off during and after power-down.

C241 is charged to 53 V through D214 in normal operation, and cannot discharge through back-biased D214 when the 53 V supply drops to zero. When the supplies drop to zero, the TTL video signal that drives Q215 also drops to zero and Q215 and Q214 are completely cut off. C241 will maintain its charge for a few minutes and keep the Cathode of the CRT at 53 V, keeping the beam turned off.

VERTICAL SECTION

The vertical portion of the video circuit board consists of two sections, a sweep generator and an amplifier.

The sweep (or ramp) generator consists of C211, C212, R227, and Q201. Capacitors C211 and C212 charge to +53 volts through resistor R227 to generate the ramp. This ramp voltage is applied to the anode of Q201, a programmable unijunction transistor. The gate of the unijunction is biased at a voltage determined by R222, R223, D201, and R224. When the anode voltage charges to the gate voltage, Q201 conducts and discharges C211 to ground through L202. As the discharge current decreases to zero volts, the unijunction stops conducting and the capacitors start to charge again through R227.

The ramp voltage is applied to the base of Darlington voltage follower Q202. The emitter voltage of Q202 is fed back to the junction of C211 and C212 to linearize the exponential ramp. Resistor R229 and Vert Linearity control R228 determine the amount of correction applied to the ramp. The amplitude of the ramp is determined by R225 and the Vert Size control, R226.

The free-running frequency of the oscillator is slightly less than 60 Hz, the normal sweep rate. Vertical sync pulses, which enter the circuit board at plug P202, pin 6, are coupled through R221, C208, and D201 to the gate of transistor Q201. The negative-going pulse lowers the gate voltage below the anode voltage and Q201 immediately conducts, discharging C211 and C212 before the free-running trip point is reached. Each succeeding sync pulse keeps the oscillator synchronized with the vertical sync signal generated by the CRT controller on the logic circuit board.

The amplifier portion of the vertical circuitry is composed of Q203, Q204, Q205, Q206, Q207, and Q208.

Under steady-state conditions, with no ramp signal

applied to the base of transistor Q204, the collector currents of Q203 and Q204 are determined by bias string R232, R233, R234, R235, and emitter resistors R236 and R241. The collector current of Q203 is nominally 3 milliamperes, and the collector current of Q204 is nominally 2 milliamperes. The difference (1 milliampere) between the two, supplies base current to driver transistor Q205, which drives output transistors Q207 and Q208. Diode D202 and resistor R244 bias transistors Q207 and Q208 so that there is enough idle current to eliminate crossover distortion. Transistor Q206 is a current source that provides base current for Q208 and the bias network, D202 and R244. The output voltage at the junction of R246 and R247 is fed back to the bias string through R239 to keep the output stable at about 25 volts.

When the ramp signal is applied to the input of the amplifier through C213, the collector current of Q204 is varied as a function of the amplitude of the ramp voltage. The difference between the currents of Q203 and Q204 drives the outputs through Q205. The output voltage is fed through C217 to the vertical deflection yoke. Resistor R248, which is in series with the yoke, generates a voltage proportional to the yoke current. This voltage is fed back to the base of Q203 (negative feedback), which changes its collector current to keep the yoke current directly proportional to the input ramp voltage.

HORIZONTAL SECTION

The horizontal portion of the video circuit board consists of three sections:

Time delay and pulse shaping.

Horizontal deflection.

High voltage supplies.

The time delay and pulse shaping circuits are triggered by the horizontal sync pulses that come from the terminal logic circuit board. They generate a time delay that provides horizontal centering and a pulse of the proper width to drive the horizontal sweep system.

The horizontal deflection system transfers energy from the power supply to the yoke in order to sweep the beam across the face of the CRT. The high voltage supplies generate the anode and grid voltages that operate the CRT.

TIME DELAY AND PULSE SHAPING

The horizontal sync pulses enter the circuit board at plug P202, pin 1. These pulses are then coupled through R249, C218, and D207 to U201. The trailing edge of each pulse triggers U201, a monostable multivibrator, causing the output (pin 3) to go high. The width of the output pulse is determined by C221, R254 and Horizontal Centering control R253. When the output pulse from U201 goes low, it triggers monostable U202. The 20 microsecond output pulse of Q202 (pin 3) is determined by C224 and R256. This pulse drives horizontal driver transistor Q216.

HORIZONTAL DEFLECTION

Transistor Q216 and driver transformer T201 drive horizontal output transistor Q217, which, in turn, drives the horizontal output transformer (flyback transformer) and the yoke. The positive-going pulse from U202 is coupled through the parallel combination of R257 and C225 to the base of Q216. During the time the pulse is high, the collector current of Q216 flows through the primary of T201. The phasing of the transformer is such that the secondary output voltage during this time is negative and keeps Q217 turned off. While Q216 is turned on and current flows through the primary, energy is stored in T201.

When the output pulse from U202 returns to zero volts, Q216 turns off, its collector current decreases to zero, the secondary voltage of T201 goes positive, and Q217 starts to conduct. The energy stored in the transformer is converted to base current and keeps Q217 turned on for the rest of the cycle. The transformer inductance, R261, R262, C227, and L203 control the base current decay and insure the best efficiency of the output transistor.

Transistor Q217 is a switch that controls the flow of energy through the deflection components. When Q217 is turned on, current flows from the power supply through the primary of horizontal output transformer T2 and through the horizontal yoke, L204, L205, and C233 to ground. During this time, the yoke current increases linearly and the beam is deflected to to the right of the screen. When it reaches the right edge, driver transistor Q216 turns on and output transistor Q217 turns off. The energy that was stored in the yoke (along with the energy stored in the primary of T2) is transferred to C229 in the form of a half-wave voltage pulse with an amplitude of 550 volts. During this cycle, the current through the yoke goes to zero and the beam returns to the center of the screen.

Capacitor C229 now discharges into the yoke, inducing a current in the opposite direction, and deflecting the beam to the left side of the screen. As the voltage across C229 decreases to zero volts, the resonant circuit of C229, the yoke, and the primary of T2 tries to oscillate in a negative direction. The energy transferred to the yoke by C229 now provides the sweep current for the first half of the scan and charges C233 through damper diode, D208.

Shortly before the beam reaches the center (before the yoke current reaches zero), transistor Q216 is turned off by U202 and Q217 is turned on. For a brief period, both D208 and Q217 are conducting in opposite directions. D208 is conducting the yoke current and Q217 is conducting the primary current of T2. Transistor Q217 turns on early to guarantee a smooth transition from negative to positive yoke current.

The value of C233 is chosen to provide "S" shaping of the current waveform through the yoke. This compensates for stretching at the left and right edges of the screen. Since the deflected beam sweeps a wider area at the edges than it does at the center for a given deflection angle, the current is decreased slightly at the left and right edges.

Width coil L204 is in series with the yoke and its reactance can be adjusted to change the total current through the yoke. If its reactance is high, the yoke current is slightly decreased and the scan width is reduced. If its reactance is low, the scan width is increased.

Horizontal linearity coil L205 is a nonlinear inductor that provides further linearity correction that cannot be provided by C233 alone.

HIGH VOLTAGE SUPPLIES

The flyback voltage pulse developed at the collector of Q217 during the horizontal retrace is rectified by D211 and C232 to provide approximately 500 volts DC. This voltage, which is filtered further by R276, C236, C237, and C238 is coupled through R4 to the CRT grid 2 (G2).

The same flyback pulse is transformer coupled to the secondary of T2 and rectified by D209 and C231 to generate a −100-volt DC supply.

Resistor R272 and Focus control R271 form a voltage divider between the +500-volt and the −100-volt supplies to provide a bias voltage for grid 4 (the focus grid). This voltage is coupled through R275 and R3 to the CRT.

Another voltage divider consisting of D212, R267, G1 control R268, R269, and D215, between the +53 V and −100 V supplies, provides a bias voltage for grid 1 of the CRT.

D215 and C242 make certain that the G1 voltage stays at some negative value with respect to the cathode to make sure that the beam stays turned off during power-down.

C242 is charged to −100 V through D215 during normal operation. When the −100 V supply decays to zero at power-down, C215 cannot discharge through back-biased D215 and maintains its charge until the CRT filament has cooled. This keeps the beam turned off and keeps it from burning a hole through the phosphor on the face of the tube.

The flyback pulse is also coupled to another secondary of transformer T2, the high voltage winding. The output pulse from this winding is about +15,000 volts. It is rectified by D1 (in the anode lead) and filtered by the internal capacitance of the CRT to provide the anode (or accelerator) voltage for the CRT.

Occasionally, the voltage stored in the internal capacitance of the CRT arcs over to the other electrodes. An arc ring built into the tube socket, and C234 (a capacitor with a parallel spark gap), in conjuction with series resistors R1, R2, R3, and R4, limit the amount of arc energy on the video circuit board to a safe value.

Flyback transformer T2 also has a filament winding that supplies 6.3 volts AC at 450 milliamperes to power the CRT filament.

TERMINAL LOGIC CIRCUIT BOARD

The terminal logic board consists of seven functional blocks:

1. Power supplies.
2. Keyboard encoder and configuration logic.
3. Processor/CPU.
4. Master clock and system logic.
5. Communications.
6. CRT and memory control.
7. Display memory, character generator, and video control logic.

The integrated circuits in each block are numbered as follows:

U401-U405	Power supplies.
U411-U419	Master clock and system logic.
U421-U429	Processor, ROM, RAM, processor control logic.
U430-U437	Keyboard encoder and configuration logic.
U451-U453	Communications and I/O drivers.
U461-U466	CRT and memory control.
U467-U479	Display memory, character generator, and video control logic.

POWER SUPPLIES

Integrated circuits U401 and U402 provide two regulated 5-volt supplies. U401 supplies 5-volts DC for the left half of the circuit board, while U402 supplies the right half of the circuit board. U403 supplies +12 volts DC, U404 supplies −12 volts DC, and U405 supplies −5.2 volts DC. These integrated circuits are internally protected against short circuits, overloads, and high temperatures. Capacitors C401, C403, C405, and C407 at the inputs of the regulators stabilize the supplies, while capacitors C402, C404, C406, C408, C409, and C440 improve the transient response of the regulators. C408 serves as the input stability capacitor for U405 and as the output capacitor for U404.

MASTER CLOCK AND SYSTEM LOGIC

Clock And Scalers

The master clock is a 12.288 MHz crystal-controlled oscillator. Crystal Y401, with C411, C413, and U411E form the oscillator. The series combination of C411 and C413 serve as the load capacitance for the crystal. U411E is the gain stage. Resistors R404 and R405 bias U411E into its linear region, while C412 bypasses any AC feedback through the two resistors. The output of the oscillator is buffered by U411D to prevent loading on the output from changing the oscillator frequency. This output is the "dot clock" and it is used by the shift register to shift dot information to the screen.

The dot clock also drives divide-by-eight scaler U412, which generates 1.536 MHz pulses. This is called the character clock. Each pulse corresponds to one character on the screen. U412 is a synchronous presettable counter that is loaded with a binary eight (1000). It counts dot clock pulses until its output reaches binary fifteen (1111). During the fifteenth

count, the ripple carry output (pin 15) goes low. This pulse, which is inverted by U411F, puts a logic one on the load input (pin 9). The next positive-going clock cycle reloads a binary eight back into the counter and the cycle repeats. The Q_C output (pin 12) generates a 1.536 MHz pulse that serves as the clock for the CRT controller. It is inverted by U466D. These two signals are referred to on the Schematic as C and $\overline{C}$. The Q_B output generates a 3.072 MHz signal that drives the clock input (pin 16) of the ACE (U451).

The dot clock also drives U413 through U475C. U413 is a divide-by-six and divide-by-two scaler. The clock drives the B input (pin 1), and the Q_D output (pin 8) generates a 2.048 MHz clock signal for CPU U421. The Q_B output (pin 11) also drives the A input. The Q_A output in turn drives the input of binary scaler U414. The output of U414 provides a 128 kHz clock (pin 6) for the keyboard encoder, U431, and a 1 kHz signal (pin 14) for the audible bell signal.

System Control Logic

The system control logic consists of I/O and memory decoding, power-up and manual reset circuits, and the bell and key click circuits.

I/O and memory decoding are accomplished by three-to-eight line decoders U418 and U419, respectively. U418 decodes address bits 5, 6, and 7 to generate eight I/O addresses:

1. Keyboard encoder ..... 200 (80_H)
2. Keyboard status ..... 240 ($A0_H$)
3. CRT controller ..... 140 (60_H)
4. Power-up configuration (primary) ..... 000 (00_H)
5. Power-up configuration (secondary) ..... 040 (20_H)
6. ACE (communications) ..... 100 (40_H)
7. Bell enable ..... 340 ($E0_H$)
8. Key click enable ..... 300 ($C0_H$)

Decoder U418 is enabled only during an I/O read or write operation to eliminate the possibility of false decoding on a refresh address coming from the Z80.

U419 decodes address bits 14 and 15 to generate three memory addresses:

1. Program ROM ..... 000.000 ($00\ 00_H$)
2. Scratchpad RAM ..... 100.000 ($40\ 00_H$)
3. Display memory ..... 370.000 ($F8\ 00_H$)

Whenever the Z80 performs a read or a write operation it will either write to or read from one of these memory or I/O addresses.

When the Computer is first turned on, the CPU, CRT controller, ACE, and keyboard control logic are cleared by the master reset signal. U415A, R407, C414, and D401 form the power-up reset circuit. When power is first turned on, C414 has no charge and temporarily holds pin 2 of U415A at logic zero. The output of U415 goes high and is inverted by U415B. The two outputs are true and the complimented reset pulses. As C414 charges through R407, it pulls the input of U415 high, turning off the reset pulses.

A manual reset can also be accomplished if you simultaneously press the Reset and right-hand Shift keys on the keyboard. U435E and U435B are connected to those keys and they drive the inputs of U415D. The output of U415D (pin 11) is coupled through R408 to pin 1 of U415A. R408 and C415 form a de-bounce circuit for the Shift and Reset keys. When the output of U415D goes low, the input of U415A is also pulled low. This generates a reset pulse.

The CPU, under the control of the ROM program, can cause a bell tone or a key click to sound through the speaker. When the CPU addresses I/O port 340, pin 7 of U418 triggers one-half of monostable U417. Its output goes low for about 200 milliseconds, causing the output of U415C to go high. This logic 1 is NANDed in U416C with the 1000 Hz signal coming from U414. The output of U416C drives speaker SP1. Diode D402 keeps the output of U416C from being driven above 5-volts at turn-off by the inductive reactance of the speaker.

When the CPU addresses I/O port 300, pin 9 of U418 triggers the other half of U417. Its output (pin 7) goes low for about six milliseconds and turns on the 1000 Hz tone. This short duration causes the tone to sound like a click.

PROCESSOR

The processor section of the terminal section consists of the Z80 processor (U421), or the CPU (central processing unit), ROM (read only memory), RAM (random access memory), and processor control logic.

Processor/CPU

The heart of the terminal logic circuit board is the Z80 CPU. It acts as a scheduling or dispatching service for the data coming into or originating from the terminal section. It examines the data it receives and determines what, if anything, it should do with it. If the data comes from the ACE (U451), for example, the Z80 will compare the ASCII word with a set of conditions determined by the ROM program, and then write the word into the appropriate memory or I/O port. If the ASCII word is a bell signal, the CPU addresses I/O port 340, and the bell tone sounds through the speaker. If the word is the letter "B", the CPU performs a memory write to the current cursor position in the display memory. If the data from the ACE is a nonvalid character or a string of characters, the CPU simply ignores the data and does nothing.

The ROM program that directs the CPU is rather long and complex, but the mechanics of the process are easy to follow. The 2.048 MHz clock signal drives the clock input (pin 6) of the CPU through U411A. This steps the CPU through an internal "machine" cycle that starts with a fetch instruction. It executes the remainder of its instructions by stepping through a precise set of a few basic instructions. These include memory read, memory write, I/O read, I/O write, and interrupt acknowledge. The basic thing to remember is that the ROM program directs the Z80 to make decisions and move data from place to place within the circuit board. Without the CPU and ROM, the decisions and data movement would have to be accomplished with hard-wired logic packages.

ROM

The read only memory, U422 and U423, is made up of a pair of 2 k by 8-bit (16384 bit) ROMs. Its eleven address inputs connect to A0 through A9 of the address bus and its eight data outputs connect to D0 through D7 of the data bus. U429A, U429B, and U416D decode the ROM select line coming from memory decoder U419 and address A11 for the CPU to generate a new address for either a 2 k or a 4 k ROM.

RAM

The random access memory for the Z80 scratchpad consists of U424 and U425, 256 × 4-bit RAMs. This scratchpad RAM provides temporary data storage for the Z80. The address inputs to each IC connect to A0 through A7 on the address bus. The lower four bits of data (D0-D3) are provided by U424; the upper four bits (D4-D7) are provided by U425. The select signal comes from U419.

Processor Control Logic

The processor requires some additional circuitry to control the interrupt process, and to provide a wait cycle for the keyboard encoder, which is slow in responding to a read cycle.

U427C is a 2-input NOR gate that monitors the INTRPT output of the ACE and pin 4 of U436B (the keyboard INTRPT). When either INTRPT output goes high, the output of U427C goes low and signals the INT input (pin 16) of the Z80 that data is available from the ACE or keyboard.

U426, U427A, and U427B form a counter that drives the $\overline{\text{WAIT}}$ input (pin 24) of the Z80. Whenever the Z80 performs an I/O read at the keyboard encoder, pin 11 of U418 drives the "reset to zero" inputs (pins 12 and 13) of U426. The A and B outputs (pins 9 and 5) of U426 drive the inputs of U427B, a 2-input NOR gate. The output of U427B holds the Z80 $\overline{\text{WAIT}}$ input low whenever the A or B outputs of the counter are high. This generates a total wait of four clock cycles (one wait cycle is automatically inserted by the Z80 on an I/O instruction) to allow the output buffer of U431 to turn on. When the C output (pin 4) of counter U426 goes high, it drives the input of U427A high. This forces the output low and turns on the clock input (pin 10) of U426. The A and B outputs now go low and the wait signal is no longer present. The Z80 then finishes up the I/O read cycle (pin 11 of U418 goes high) and the counter is reset to zero and held there until the next keyboard read.

U428 provides a nonmaskable-interrupt ($\overline{\text{NMI}}$) that operates under the control of the ROM program. The NMI routine is used when the program wants to read something into the display memory during the vertical blanking period. The data input of U428A is driven by A2 of the address bus. The T, or clock, input is driven by the complemented CRT controller I/O select that comes from pin 12 of I/O decoder U418. U466C provides the complemented signal. When the

program wants to write during vertical retrace, it addresses the CRT controller while holding A2 high. The Q output of U428A is clocked high and drives the reset input of U428B high. The vertical sync signal drives the T input of U428B and clocks the $\overline{Q}$ output low as soon as the sync signal begins. The $\overline{NMI}$ input of the Z80 goes low and the program immediately jumps to the "write during retrace" routine. Part of that routine will write a zero to the data input of U428A to clear the NMI signal.

KEYBOARD ENCODER AND CONFIGURATION LOGIC

Keyboard Encoder

The keyboard of the Computer consists of single-pole, single-throw switches in a matrix that is scanned by keyboard encoder U431. Outputs X1 through X9 go high, in sequence, and drive one of the Y1 through Y10 inputs if one of the switches is depressed. The encoder uses the X and Y information to generate a unique binary code for each matrix intersection, and this code is latched internally when a key is depressed. The encoder generates a data strobe (DS), which comes from pin 13 of U431, for each key closure. It also clocks the T input (pin 3) of U432A and the $\overline{Q}$ output of U432A goes low. The $\overline{Q}$ output drives and input of U436B. The output of U436B generates an $\overline{INT}$ signal, which is coupled to the Z80. When the Z80 services the interrupt at I/O port 200, pin 11 of U418 clears U432A (through U436C and U435A) and the INT signal is defeated. Pin 36 of U431 is also a binary data output and it is latched in U432B by the I/O read at 200. The keyboard interrupt routine also checks the keyboard status in another I/O read operation. The keyboard status check reads the state of the following:

1. Control key.
2. Shift keys.
3. Repeat key.
4. Break key.
5. Off-line key.
6. Caps Lock key.
7. Data Strobe.
8. Data bit latched in U432B.

The ROM program uses this information in conjunction with the encoder data to determine the routing of the data within the Computer. Pin 10 of I/O decoder U418 drives enable inputs (pin 19) of buffers U433 and U434 to put the status information on the bus. The Caps Lock, Break, Off Line, Control, Repeat, and Shift (left) keys are connected directly to the inputs of these buffers. The outputs of U432A and U432B are also connected to the inputs of the buffers.

The binary data outputs of the keyboard encoder drive the address inputs (A0-A7) of ROM U430. U430 converts the binary data from the keyboard encoder to ASCII data. The data outputs of U430 drive the D0-D7 bits of the data bus. The chip select input (pin 18) of U430 is driven by pin 11 of U418 (the I/O decoder).

When the Repeat key is held down, the input of U435D is low and its output is high. This enables the repeat rate oscillator, U436A, R414, and C418. The repeat frequency, which is approximately 15 Hz, is determined by R414 and C418. When the Repeat key is released, the output of U435D goes low, forcing the output of U436A high and disabling the repeat function.

The two shift keys are ORed together in U436D. Its output drives the shift input (pin 21) of U431. When the Control key is typed, it forces the output of U435F high, which drives the control input (pin 19) of keyboard encoder U431.

Configuration (Power-up) Logic

When the system is first turned on, the ROM program must program the ACE for the baud rate and parity that you selected on switches S401 and S402. The program addresses I/O port 000 and pin 15 of I/O decoder U418 drives enable inputs of U433 and U434 to put the information selected by the switches on the bus. The program then interprets the data and configures the ACE accordingly. I/O address 040 is used in a similar manner. Pin 14 of the I/O decoder (U418) enables buffer U437 and puts the data from S402 on the bus.

COMMUNICATIONS AND I/O DRIVERS

The Computer talks to the outside world through an Asynchronous Communications Element (ACE) and EIA RS-232C compatible line drivers and receivers. The ACE (U451) converts parallel ASCII data to serial data and drives the communications line through line

driver U452. The ACE also converts serial data coming from line receiver U453 into parallel ASCII data. The ACE puts this data on the bus when the ROM program requests it.

ACE/UART

U451 is an Asynchronous Communications Element that performs the following functions:

1. Converts data from parallel to serial and vice versa.
2. Divides a master clock frequency by a programmable divisor to generate a desired baud rate.
3. Programs the data characteristics, parity, stop bits, and character length.

The characteristics of the ACE must be programmed into the internal registers of U451 by the ROM program through the address and data busses. Bidirectional data bits (pins 1-8) of U451 connect to the system data bus. The address inputs (pins 28, 27, and 26) connect to the system address bits A0, A1, and A2. When the ROM program addresses I/O port 100, pin 13 of I/O decoder U418 selects the $\overline{CS2}$ input (pin 14) of the ACE. The Z80 can then read or write data by enabling the data input and data output strobes at pins 21 and 18 ($\overline{DISTR}$ and $\overline{DOSTR}$).

When the ACE receives a complete serial word from the EIA interface, it signals the Z80 that there is data available by pulling the Z80 $\overline{INT}$ input (pin 16) low. The Z80 then examines the internal status and data registers of the ACE, reads the data word, and routes it to the proper device within the terminal section.

For a further description of the ACE and how to program it, refer to Page 14-1 in the "Appendix."

I/O Drivers

The standard EIA interface communicates by means of a serial stream of voltage levels that correspond to logic ones and zeros. A logic one (or mark) on the data lines is a voltage between −5 and −15 volts. A logic zero (space) on the data lines is a voltage between +5 and +15 volts. On the control lines ($\overline{DTR}$, $\overline{RTS}$, $\overline{RLSD}$, $\overline{DSR}$, $\overline{CTS}$), a voltage between +5 and +15 volts is considered to be ON, and a voltage between −5 and −15 volts is considered to be OFF.

U452 is a standard EIA line driver. A logic one on the input of U452C drives the serial data outline to an EIA logic one, or "mark". A zero on the input forces the line to an EIA zero, or "space." U452B and U452D drive control lines $\overline{DTR}$ (Data Terminal Ready) and $\overline{RTS}$ (Ready To Send) in a similar manner.

U453 is a standard EIA line receiver. The serial data in line drives the input of U453A, which converts the EIA voltages to TTL levels and drives the serial input of the ACE. Likewise, the $\overline{RLSD}$, $\overline{DSR}$, and $\overline{CTS}$ line signals drive the inputs of U453B, U453D, and U453C, respectively. The outputs drive the appropriate control inputs of the ACE.

CRT AND DISPLAY MEMORY CONTROL

The heart of the video logic system is the CRT controller. This device generates all of the sync and blanking signals and display memory addresses for the video system. The memory control is used to select either the address coming from the CRT controller or the address bus, and to synchronize read and write pulses.

CRT Controller

The CRT controller, U461, is a fully programmable device that is set up by the ROM program during power-up. Its bidirectional data bits (pins 33-26) connect to system data bits D0-D7. Its address or programming inputs come from the following four input pins:

Pin 22. Read/Write (R/$\overline{W}$) – Determines whether the controller's internal register file is to be written to or read from. A write is a logic zero.

Pin 24. Register Select (RS) – Selects either the address register (RS=0) or one of the data registers (RS=1) of the internal register file.

Pin 25. Chip Select ($\overline{CS}$) — A zero sets the CRT controller to read or write the internal memory file.

Pin 23. Enable (E) — Enables the I/O buffers and clocks data to and from the CRT controller. Data is clocked on the falling edge of the enable signal.

The internal registers are written to or from by means of the address register. The Z80 sets up the programmable registers by first writing a register number into the address register when the register select input is low. It then performs a write operation when the Register Select input goes high.

Each of the CRT controller's registers is programmed at power-up with appropriate data to generate the SYNC, timing, and refresh signals. The memory address outputs (MA0-MA10) drive the address of the display RAM through multiplexers U463, U464, and U465. The scan row address outputs (RA0-RA3) drive the address inputs of character generator U473. The display enable output (EN) is a logic one whenever the CRT controller, U461, is addressing a port of the RAM during the time it should be displayed. This serves as a blanking output whenever it is a logic zero. The cursor output goes to a logic one when the RAM location being addressed is equal to the address stored in the cursor address registers.

Controller Read/Write Logic

The $\overline{CS}$ and EN inputs of the CRT controller must be selected in a particular sequence to perform read and write operations to and from the controller. The enable input pulse (pin 18) must always be inside the CS pulse. When the I/O request for address 140 appears at pin 12 of U418, the clear input of U462A goes low, and the $\overline{Q}$ output immediately drives the CS input low. The $\overline{Q}$ output drives the data input of U462B to a logic one. At the same time, U466C puts the Clear input of U462B at a logic one.

The next CPU clock pulse at pin 11 of U462B clocks the logic one at the data input through to the Q output. This delays the leading edge of the enable pulse until approximately one clock cycle after the leading edge of the $\overline{CS}$ pulse. When the I/O request at address 140 goes away (returns to logic one), the output of U466C immediately clears U462B and its Q output drives the EN input of the CRT controller to zero. The clear input of U462A goes high at the same time, but the Q output remains low until the next CPU clock pulse at pin 3 clocks the logic one at the data input through to the output, terminating the $\overline{CS}$. This delays the trailing edge of the $\overline{CS}$ pulse until after the trailing edge of the EN pulse.

Display Memory Control

The display memory control consists of an address bus multiplexer, a bidirectional bus buffer, and some gates that control the display memory write enable (WE) and chip select ($\overline{CS}$) inputs.

The address bus multiplexer consists of quad 2-input multiplexers U463, U464, and U465. Their select inputs are tied together and controlled by memory decoder U419. When no read or write operations are being performed on the display memory, the select inputs are at a logic one, and the memory addresses (MA0-MA9) generated by the CRT controller drive the address inputs (A0-A9) of RAMs U467-U470. When the Z80 addresses the memory, pin 9 of U419 pulls the select input to a logic zero, CRT controller memory addresses MA0-MA9 are disconnected from the display RAM, and address bus bits A0-A9 are connected to the RAM address inputs. The Z80 can then read from or write into the display RAM.

Bus buffer U471 isolates the main data bus from the secondary or refresh, bus. During the screen refresh period, the data outputs of the display RAM drive the data inputs of the character generator continuously. This would prevent the processor from having access to the bus except during retrace times. However, by isolating the refresh bus from the main bus, the Z80 can have continuous access to the main bus and the display RAM and character generator can have continuous access to the secondary bus (refresh bus). When the Z80 needs access to the display RAM, it addresses the memory, which enables U471 through pin 9 of U419, and connects the main bus directly to the secondary bus.

U466A and U466B provide the $\overline{CS}$ signals for the display RAMs. During the screen refresh cycle, pin 11 of U465 is driven by MA10 and pins 1 and 4 of U466 are logic one. The output of U466A provides the $\overline{CS}$ for RAMs U469 and U470, and drives input pin 5 of U466B. The output of U466B is the complement of the $\overline{CS}$ signal and it drives the $\overline{CS}$ input of RAMs U467 and U468. During a display RAM read or write cycle, pins 1 and 4 of U466 are driven by the $\overline{R/W}$ signal coming from pin 3 of U416A. This eliminates the possibility of a contention problem on the secondary (refresh) bus between the display RAMs and buffer U471.

The write (WE) inputs of the RAMs are connected together and they are controlled by U429C. The WE (pin 8 of U429C) cannot go low unless pin 9 of U419 is low (memory is selected) and the Z80 WR output is low.

DISPLAY MEMORY, CHARACTER GENERATOR, AND VIDEO CONTROL LOGIC

This section of the terminal logic circuit board essentially runs by itself (in conjunction with the CRT controller) after being programmed by the Z80. The CRT controller continually provides refresh addresses for the display RAM, while the output of the RAM continually provides data for the character generator and the video shift register.

Character Generator

Character generator U473 is a 2048 × 8 (16384 bit) read only memory that converts the ASCII data stored in the display memory into dot information for the video shift register. Address inputs A0-A3 (pins 5-8) are driven by the scan row address outputs of the CRT controller (RA0-RA3) to select a particular row of dots within a character space. Address inputs A4-A10 connect to the secondary data bus through 8-bit latch U472. These inputs use ASCII data to address the dot data stored in the ROM. The data outputs (01-08) of U473 supply video dot data to the parallel inputs of video shift register U474.

The inputs of 8-bit latch U472 connect to the secondary data bus. Data bits D0-D6 are latched into U472A and drive the character generator. Data bit D7 is the reverse video bit. It is latched in U472H and drives an input of U479A.

Video shift register U474 latches parallel dot data from the character generator at inputs A-H and shifts it out of output Q_H in synchronism with the dot clock (the dot clock drives the clock input, pin 7). The shift register is loaded (the dot data is latched) on a positive-going transition of the dot clock while the shift/load input is held low by the character clock coming from pin 12 of U411F. The dot data at input H appears immediately at output Q_H. The next leading edge of the dot clock shifts the data that was latched at Q_G. The next edge of the dot clock will shift the data that was latched in Q_F, and so on. After the data from Q_A is shifted to the Q_H output, the load input goes low, and the next character cycle begins.

Video Control Logic

The video control logic consists of two sections: a series, or chain, of gates and latches associated with video, cursor, and reverse video data; and a chain of gates and latches associated with blanking data.

The display enable (blanking) and cursor data from pins 18 and 19 of the CRT controller is coincident with MA0-MA10, which address the display RAM. The display enable bit is latched in U476D (after passing through AND gate U477C) by the complemented character clock pulse coming from pin 11 of U466D. The cursor bit is latched in U476F. This delays the two signals by one character time. They are delayed for one more character time by being latched in U476E and U476G, respectively. The two character delays are necessary to compensate for the delays in the display RAM/character generator "pipeline".

When MA0-MA10 address the RAM, it takes approximately 450 nS for the data to be valid at the outputs. Once it settles down, the next character clock latches it in U472. The data at the output of U472 then addresses character generator U473. The data at the output takes another 450 nS to settle, and it is latched in the shift register by the following clock pulse. (Since the character clock pulses are 650 nS apart, the RAM and character generator outputs have plenty of time to settle.) This two-character delay matches the delays for the cursor and display enable, so that everything is synchronized.

The reverse video bit (D7) in the display RAM is latched first in U472, and then in U476H (after passing through AND gate U477A), so that it too arrives coincident with the video, blanking, and cursor data.

The video dot data from pin 13 of the video shift register and the cursor data coming from pin 16 of U476G are exclusive-ORed in U475A. This causes the cursor dots to reverse when the cursor happens to be coincident with video information, and keeps the cursor from disappearing when it occupies the same space as a character.

The video/cursor information coming from pin 3 of U475A is then exclusive-ORed in U475B with the reverse video data coming from pin 19 of U476H. When pin 19 is logic zero, the video/cursor data passes through U475B just as it is entered. If pin 19 of U476H is logic one, the data is reversed, and the character appears on the screen as black dots on a

white background. The reverse video function can be disabled under the control of the ROM program when a logic zero is written into latch U478B via the address bus. Address bit A3 drives the data input of U478B, and its clock input is clocked when the CRT controller is addressed (I/O address 140). If the reverse video is to be defeated, the Q output (pin 9) of U478B puts pin 1 of AND gate U477A at a logic 0 and disables the reverse video bit coming from pin 19 of latch U472.

The video/cursor/reverse data coming from pin 6 of U475B is ANDed in U477D with the display enable data coming from pin 12 of U476E. If the display enable is logic 1, the video data goes to the video circuit board; if it is a logic zero, the video is blanked.

When the Z80 performs a read or write operation on the display RAM, it disturbs the pipeline, and the data on the secondary (refresh) bus does not coincide with what should be written on the screen. Consequently, the video is blanked during a read or write. When pin 9 of memory decoder U419 goes low to select the display RAM, it also drives the clear input (pin 1) of U476. The Q output (pin 6) of U476C drives pin 9 of AND gate U477C to a logic zero and disables the display enable. At the same time, the Q output (pin 12) of U476E drives pin 12 of AND gate U477D to a logic zero, blanking the video information coming from the video chain. The screen will blank as long as the RAM is selected.

When pin 9 of U419 goes high to deselect the RAM, U476 is no longer held cleared. The logic one at the D input of U476C is clocked through to its Q output on the next character clock pulse, and it is clocked through U476D and U476E on the next two pulses. This three-character delay gives the pipeline time to reload with valid information before the video is enabled.

The propagation delays through the various gates and latches (U474, U475A, U475B, and U476) from the edge of the character and dot clocks to their various outputs is not always constant, so another delay is used. Latch U478A acts as a mini-pipeline, clocked at the dot rate. The data input to U478A is the composite video/cursor/reverse/blanking data, and its T input is clocked by the dot clock. This 80 nS delay lets all data settle to valid states before it is sent to the video circuit board.

The sync and video signals are buffered before they leave the Terminal logic circuit board. U479A inverts and buffers the video data. U479B inverts and buffers the vertical sync signal coming from pin 40 of the CRT controller. U479B buffers the horizontal sync signal coming from pin 39 of the CRT controller.

CPU LOGIC CIRCUIT BOARD

Refer to the CPU Logic Circuit Board Block Diagram (Illustration Booklet, Page 15) as you read the following material.

SYSTEM CLOCK

Crystal X501, in conjunction with U501A, form a crystal oscillator which operates at 12.288 MHz. Capacitors C501 and C503 provide the load for the crystal, while R501 and R502 force U501A to operate in its linear mode. C502 acts as a low-pass filter to insure that the crystal will determine the operating frequency of the circuit rather than the delay time of U501A.

U502 operates as a divide-by-6 divider with a symmetrical output and, as such, provides the 2.048 MHz clock signal. U503 operates as a divide-by-1024 divider and provides the 2 mS clock signal.

POWER-UP AND RESET

When power is first applied, capacitor C506 is discharged. As the +5 V rail becomes active, C506 begins to charge. Approximately 150 mS after the +5 V rail reaches +5 volts, pin 6 of U508B goes low and pin 8 of U501D goes high and terminates the power-up reset operation.

As long as the reset and shift keys are held down on the keyboard, pin 3 of U506A is held low and the display electronics are continuously reset. However, the Computer is not reset until the keys are released. This insures that the display will complete its reset function before the Computer resets; thus resulting in a proper indication on the display.

The rising edge of the KBRST L signal toggles U506A and causes its Q output (pin 5) to go high. This signal is coupled to an 8 μS one-shot (U507A) by U508A, and is the trigger signal for the one-shot. U507A's output is the reset signal used by the Computer and is coupled to the Computer by U508B and U501D.

U508A and U508D form an R-S flip-flop which guarantees that a reset will occur only during the op code fetch portion of an M1 cycle. Therefore, the refresh for dynamic memories will not be disturbed and no information will be lost.

CENTRAL PROCESSING UNIT (CPU)

The CPU is a Z-80 microprocessor which runs at 2.048 MHz. Pictorial 9-1 shows the timing during an M1 cycle (instruction fetch). Notice that refresh occurs at the end of the instruction fetch. Pictorial 9-2 shows the timing of a memory read and a memory write cycle, Pictorial 9-3 shows the timing during an I/O cycle, and Pictorial 9-4 (on Page 9-18) occurs only at the end of an instruction cycle and that both an M1 and an IORQ are generated.

CONTROL LOGIC

U509 is a non-inverting buffer, while U562D complements BM1 L to produce BM1 H. U515A OR's the I/O request and memory request signals to produce the latch address signal (LA L). U562A and U562B form a 100 nS delay for the memory request signal; thus DMERQ L is the delayed memory request. This delay occurs only on the leading edge and not the trailing edge of the signal.

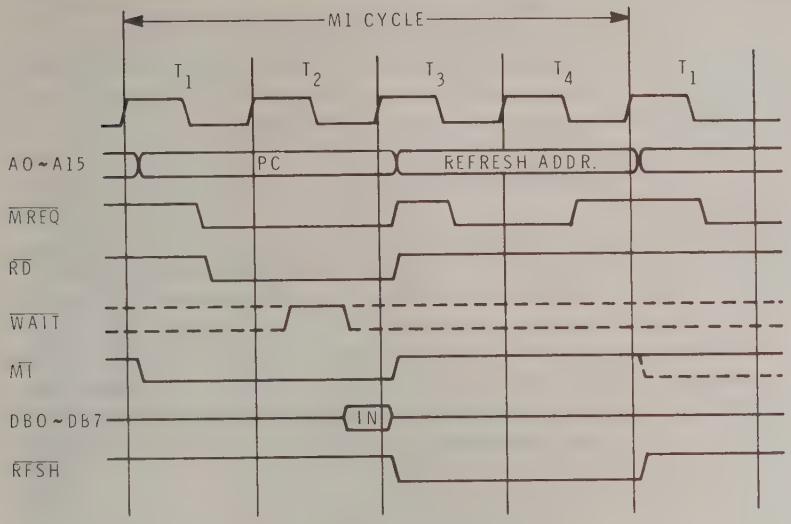
ADDRESS LATCH

U510 and U511 are transparent latches. Any data at their inputs is transferred to their outputs when LA L is high. When LA L is low, the last data pattern to occur just before the LA L high-to-low transition is retained at their outputs.

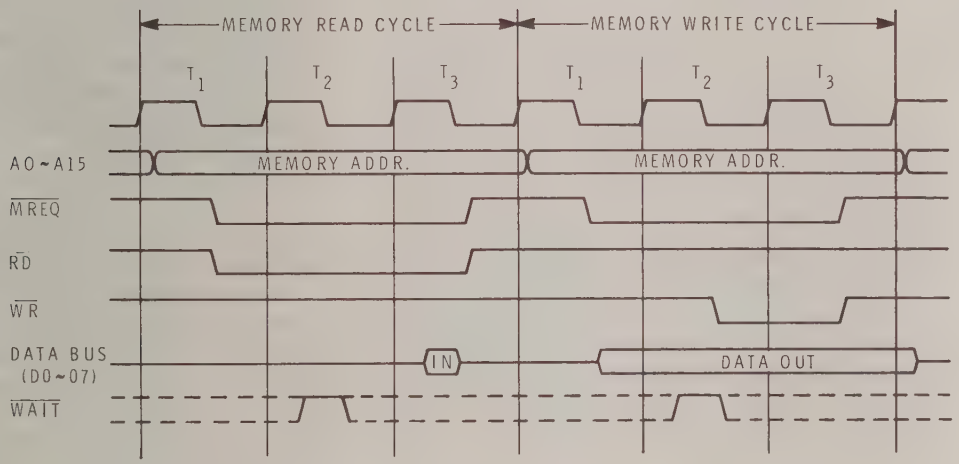
MEMORY MAP DECODER

U517 and U516 are PROMs which decode the latched address lines and determine which memory chips are to be enabled. Specifically, U517 determines which of the eight 8k banks is to be enabled, and U516 determines which of the 1k pages within bank 0 is to be enabled. However, the address lines are not the only criteria used to determine memory selection.

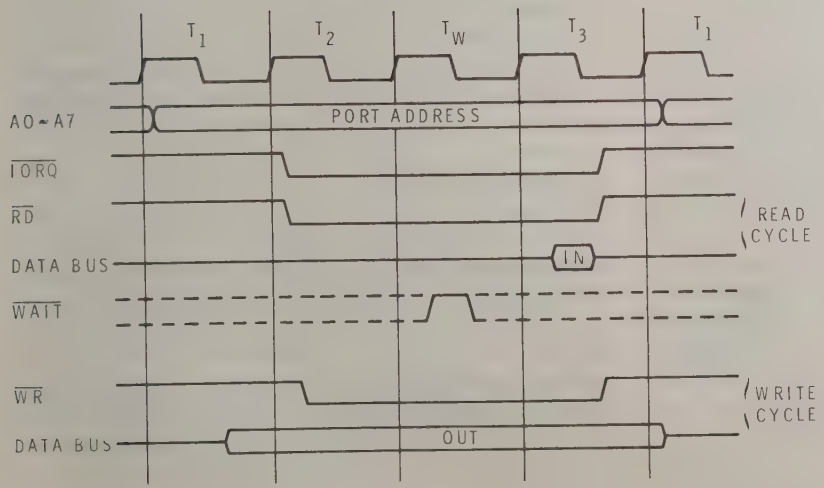
Jumpers JJ501, JJ502, JJ503, and JJ504 select various memory configurations. That is, if only 16k of user RAM is installed, then JJ501, JJ502, and JJ504 are jumpered for 0 and JJ504 is a "don't care". As a result, only RASO L will be selected (coincident with the selected address being within banks 1 and 2). At no time will RAS1 or RAS2 be selected. In addition, NOMEM L will be asserted low whenever banks 3, 4, 5, 6, or 7 are accessed. This results in U521 being enabled and all 0's being forced onto the data bus



PICTORIAL 9-1



PICTORIAL 9-2



PICTORIAL 9-3

(required by some software memory sizing routines). If, however, 32k of user memory is installed, JJ501 and JJ503 will be jumpered for 0 and JJ502 jumpered for a 1. JJ504 is again a "don't care". Now, when banks 3 and 4 are addressed, RAS1 L will be asserted instead of NOMEM L, while RAS0 L requires the same conditions as before.

The write enable line (WE L) will be asserted low whenever there is no read (BRD L is high), no refresh (BRSH L is high), memory is selected (DMERQ L is low), and the bank selected contains writeable memory. Thus, since NOMEM L contains no writeable memory, WE L cannot be asserted if NOMEM L is asserted. However, bank 0 contains both writeable and non-writeable memory. Therefore, WE L will be asserted for bank 0 if the first three conditions are met regardless of the fourth condition. It is up to U516 to either enable or not enable the page select lines (MC5001 L, MC5023 L, MC505 L, and MC5067 L), depending on whether the memory addressed is writeable or not.

Write enable line 0 (WE 0 L) is used with the 1k of RAM associated with the floppy disk. It is only asserted when WE L is asserted, MC505 L is asserted, and FMWEN H is asserted. It is controlled by bit 7 of I/O port 177.

If the refresh line (BRFSH L) is asserted low, then WE L is not asserted. In addition, RAS0 L, RAS1 L, and RAS2 L will be asserted (consistant with the position of jumpers JJ501, JJ502, JJ503, and JJ504) regardless of the bank address. This insures that all applicable dynamic RAM will be refreshed.

DYNAMIC RAM MULTIPLEXER

U513 and U514 multiplex the 14 address lines required for 16k of memory into the 7 lines available on the memory chips. U512A is used to clock the row and column addresses into the memory chips in the proper sequence and at the proper time. Assume that a memory cycle has just begun (DMERQ H is high). At this time, the row addresses are multiplexed onto the memory address lines (MA0-6 L) and the applicable RAS line (RAS0-2 L) is asserted low; thus, latching the row address into the memory chips. One half clock cycle later, U512A is toggled. This causes the row addresses to be removed from the memory address lines (MA0-6 L) and the column addresses to be multiplexed in. A short delay later (determined by R513 and C519), the CAS L line is asserted and the column address is latched into the memory.

The multiplexer will be returned to its normal state (RAS address selected) when either a refresh cycle is initiated (clears U512A) or after the first clock cycle following the memory cycle (DMERQ H is low, causing the Q output of U512A to go low).

SYSTEM ROM

U518 is the system ROM which resides in the bottom 2 pages of memory (0, 1). An additional 2k of ROM can be added to the system at Pages 2 and 3 by installing jumper JJ508 at "A". Jumpers JJ505, JJ506, and JJ507 are used to switch between the 3-voltage EPROMs and the single voltage ROMs. When the jumpers are installed in their 0 locations, then the 3-voltage EPROMs are used.

FLOPPY DISK ROM

U520 is the floppy disk ROM which resides in the address map at Pages 6 and 7. It is restricted to a single voltage ROM. However, by moving jumper JJ508 to 'B', U519 will serve as the floppy ROM and a 3-voltage EPROM can be installed. It is now no longer possible to use 4k of system ROM.

FLOPPY DISK RAM

U523 and U525 are used by the floppy disk and reside in Page 5 of bank 0 of the memory map. They are organized as 1k × 4 static RAMs. Provisions are made for an additional 1k of static memory at Page 4. However, it cannot be write protected as the memory at Page 5 can.

SYSTEM RAM

The system RAM is organized as 16k × 1 dynamic RAMs and reside in banks 1, 2, 3, 4, 5, and 6. U526 through U533 comprise banks 1 and 2, U534 through U541 comprise banks 3 and 4, and U542 through U549 comprise banks 5 and 6. Bank 7 is reserved for ROM.

I/O MAP DECODER

U550 is a 256 × 8 PROM, and decodes the various I/O ports required by the Computer. I/O 362 L is the general purpose port. I/O NMI L is used by the system to trap all accesses to ports 360, 361, 372, and 373. This enables the system to run previous software developed for the H8 Computer. That is, accesses to the H8 front panel are rerouted to the system console.

Since the interrupt acknowledge generates a BIORQ L signal, this could cause an I/O request to I/O 362 or I/O NMI. The effect would be to either cancel the interrupt before it could be processed or to generate an NMI request. In either case, the Computer will get lost. Therefore, since the interrupt acknowledge also generates an M1 signal, M1 is used to deselect U550 during the interrupt acknowledge cycle.

INTERRUPT LOGIC

U557 detects that an interrupt request has occurred. It transmits this information to the processor by the INT L line. The interrupt priority is also determined by U557, and is available on output pins 6, 7, and 9. This information is transmitted to the data bus during the interrupt acknowledge cycle by U558.

The highest priority is interrupt level 5, while the lowest priority is the 2 mS clock at level 1. In processing the interrupt, the CPU is operating in the 8080 mode. Therefore, the data on the data bus at acknowledge time is an instruction (U558 incodes the data from priority encoder U557 such that the processor sees a restart instruction). This instruction directs the processor to execute the instruction at ROM address 10 for the 2 mS clock, at address 20 for the single step, at address 30 for the INT 3L, etc.

SINGLE STEP AND 2 mS CLOCK

The 2 mS clock is controlled by U506B. It is enabled by writing a '1' on data line D1 H at I/O port 362. Once enabled, the next positive transition of the 2 mS clock will trigger U506B and cause its output (pin 8) to go low; thus enabling interrupt level 1. The clock handler must, as part of its routine, disable the clock interrupt (clear U506). Otherwise, another interrupt will be generated as soon as an EI instruction is executed. This occurs because U506B is operating as a latch. It will be cleared whenever a write to I/O port 362 occurs. If D1 H is low when this write occurs, then the 2 mS clock will be disabled. If D1 H is high when the write occurs, then U506B will only be cleared and the 2 mS clock will still be enabled; thus allowing another interrupt to occur at the end of the next 2 mS period. At power-up or keyboard reset, U506B is enabled. Consequently, a 2 mS clock interrupt will occur immediately after an EI instruction has been executed.

The single step is enabled by D0 H and I/O port 362. With bit 0 of port 362 low, U555A and U555B are held in their initialized state. When bit 0 of port 362 is high, the single step is enabled.

U556A synchronizes the 2.048 MHz clock with the M1 cycle and valid data on the data lines. U515B and U554 decode the EI instruction for U555A. Thus, an EI instruction causes the D input of U555A (pin 2) to be asserted low. U555A is then toggled by U556A; thus, setting the T input of U555B (pin 11) low. At the end of the next M1 cycle, U555B is toggled, which latches its output (pin 9) low. Another M1 cycle is executed, which now toggles U556's output (pin 8) high. This generates the interrupt at level 2 (restart 20). The software sequence is:

1. Enable single step
2. Wait for keyboard
3. EI instruction
4. RTI instruction
5. Execute single program instruction
6. Interrupt out of program
7. Disable single step
8. Enable single step (step 7 and 8 are required to reinitialize single step logic)
9. Wait for keyboard
10. Etc.

GENERAL PURPOSE PORT

The general purpose port is located at I/O address 362, and is comprised of U551, U552, U553B, and DS1. A read from this port puts the dip switch (DS1) status on the data lines. A write to this port controls the single step and 2 mS interrupt logic. In addition, 4 other lines (MEM0 H, MEM1 H, IO0 H, and IO1 H) are available. These lines are routed to the accessory connectors on the CPU logic circuit board and are not presently used.

CONSOLE SERIAL PORT

This port, at I/O locations 350-357, is used to communicate with the console terminal. U559 and U560 convert the TTL levels from and to the ACE (U561) into standard EIA signals. U515D acts only as an inverter for the reset line, and Q1 provides both inversion and the WIRED-OR function for interrupt level 3 (restart 30).

The clock for the ACE is supplied by logic inside U561, and is crystal controlled by X502. C522 and C523 provide the load for the crystal, and R515 and R516 provide the proper bias for the internal devices. The clock thus generated is routed to the I/O accessory connectors for use by the Serial Interface Accessory circuit board.

For a description of the ACE and how to program it, refer to Page 14-1 in the "Appendix."

ACCESSORY CIRCUIT BOARDS

SERIAL INTERFACE CIRCUIT BOARD

The first port on this circuit board is located at I/O 340-347 and is used as the line printer port. Its output is standard EIA with a DCE connector. The second port is located at I/O 330-337 and, again, is used as the standard EIA output. However, this port is terminated with a DTE connector for communication with a MODEM. The third port is located at I/O 320-327 and is configured for EIA with a DCE connector. All three ports can be jumpered for interrupt levels 3, 4, or 5 and use the 1.8432 MHz clock generated by the console serial port.

The main logic device for each port is the 8250 ACE. For a description of this device and its programming instructions, refer to Page 14-1 in the "Appendix."

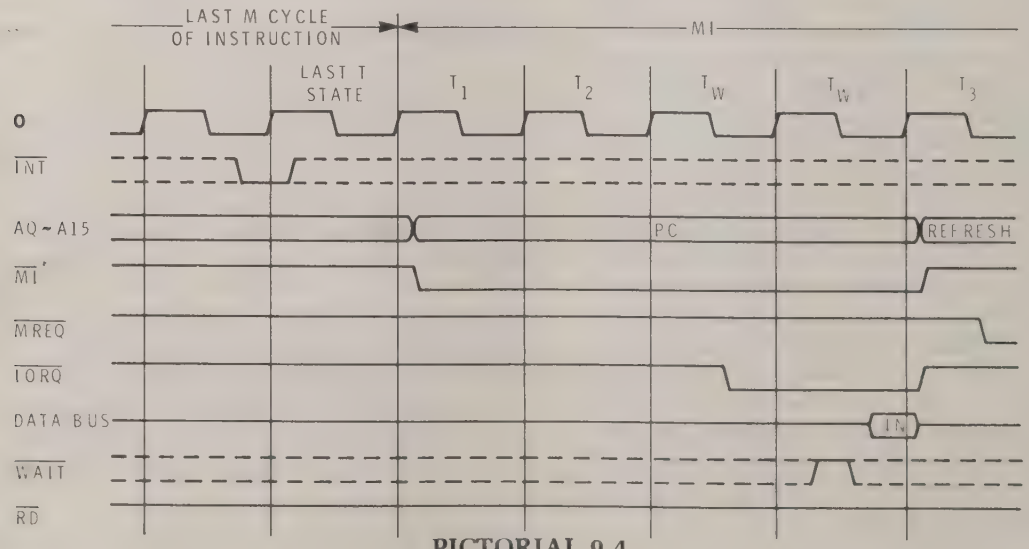
CASSETTE INTERFACE CIRCUIT BOARD

U707 converts parallel data coming from the data bus to the NRZ (non-return-to-zero serial data) format for writing to the cassette tape machine, and converts NRZ format data to parallel data for reading from the cassette machine. However, although the cassette handles the data in a serial format, it is not NRZ. The actual cassette is tone encoded with two tones: one at 2400 Hz which corresponds to a logic '1', and one at 1200 Hz which corresponds to a logic '0'. U704A, U708B, U708C, U709, and U710 convert the tone encoded data from the cassette to the NRZ format. U715A, B, C, and D change NRZ data to tone encoded data.

U713 and U714 divide the 1.8432 MHz clock into the two tones required by the tone encoder, while U706C buffers the 1.8432 clock signal as it enters the circuit board. The 1200 Hz (pin 13 of U714) clocks the NRZ data from U707. (See Pictorial 9-5, Illustration Booklet, Page 12.) This data is available on pin 19 of U707 and turns gates U715C and U715D on and off as required. When U715C is on, 2400 Hz (logic 1) is recorded; and when U715D is on, 1200 Hz (logic 0) is recorded. The timing is such that two cycles of 2400 Hz are recorded for each logic '1' bit, and one cycle of 1200 Hz is recorded for each logic '0' bit. The outputs from U715C and U715D are OR'ed by U715B and used as the composite tone encoded output.

The tone encoded data from the tape is clamped by diodes D704 and D705; after which U711D amplifies and shapes it into TTL level signals. The signal is now ready to be decoded into NRZ data.

Refer to Pictorial 9-6 (Illustration Booklet, Page 12) for the following discussion of the decoding. One-shot U709 has a timing period of 521 μS, and clocks each bit of the NRZ data into U707 on its rising edge. One-shot U710 has a timing period of 313 μS, and toggles U704A through U708B. U708B inverts U710's output so that U704A toggles when U710 times out. Due to the timing of U710 with respect to the '1' and '0' tones, the 'D' input of U704A will be low for the 2400 Hz tone when U710 times out and high for the 1200 Hz tone. The NRZ data is thus reconstructed at U704A's output (pin 6).



PICTORIAL 9-4

The above technique requires that U707 operate in the X1 mode for serial data transfers. This is done by forcing the X1 whenever U707 requests a mode word. After reset, the first operation which must be performed is to set U707 by writing in the mode word. The reset forces U704B's output to the following states: pin 9 low and pin 8 high. Consequently, U702C's output (pin 8) is low (BRD L is high since it is a write operation) and U702B's output (pin 6) is high. Therefore, U703A and B are enabled and U705A and D are disabled so that an X1 is written into U707. When the write operation terminates, U704B is toggled; thus enabling U705A and U705D and disabling U703A and U703B. The data bus is now permitted to arrive at U707 unmodified. If bit 6 is set in the command word, then U707 is forced into the mode word state. U704B is also returned to the mode word state since D6 H is asserted and AND'ed with pin 9 of U704B to force the 'D' input of U704B low if the previous state was the command state. Completion of the write operation toggles U704B such that its outputs are the same as if a reset had been executed. A read operation allows data to be transferred to the data bus unmodified and also disables U703A and U703B, and U705A and U705D.

Relay K702 turns on the tape recorder for the playback mode, and K701 turns on the recorder for the record mode. The outputs of the two relays can be jumpered together by JP701 and JP702 for single recorder operation. When U707 turns K701 on, pin 1 of U711 goes high. This causes the output (pin 10) of the U712 to turn off; thus allowing C1 to charge toward +5 volts through R1. When C1 reaches +2.75 volts (in approximately three seconds), U711's output (pin 14) goes low. Since the software is testing for this, a 3-second wait after recorder turn-on occurs before any data is recorded.

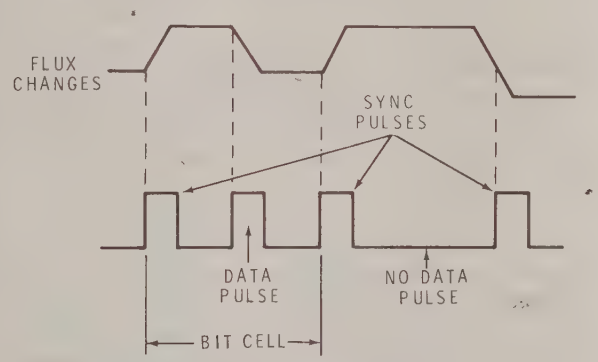
FLOPPY DISK INTERFACE CIRCUIT BOARD

The floppy disk uses a storage format similar to a 45 rpm record. Each diskette is divided into 40 concentric tracks, with the outermost track labeled track 0 and the innermost track labeled track 39. This is shown in Pictorial 9-7 (Illustration Booklet, Page 13). Each track is further divided into ten sectors. Each sector stores 256 bytes of data plus checksum and address information to prevent data errors. The check character allows the system to be confident of the data integrity. The address information confirms that the head was moved to the desired track and that the correct sector has been found for a read or write operation.

There are eleven holes spaced around the center of the diskette for sector and index identification. Ten of the eleven holes are evenly spaced to indicate the start of a sector. The eleventh hole (the index hole) is spaced halfway between the sector 9 hole and the sector 0 hole. This uneven spread is recognized by the controller and informs the controller that the next sector hole corresponds to sector 0. This method of sectoring is known as hard sectoring. Since the diskette rotates at 300 rpm, the index hole is encountered every 200 milliseconds.

Information is stored on the diskette as serial magnetic flux changes. To prevent variations in rotational speed from impairing data accuracy, each serial bit is bracketed by a synchronizing bit. (See Pictorial 9-8.) A data pulse occurs if the data bit equals one and does not occur if the data bit equals zero. As mentioned previously, the data is recorded as a flux reversal. This is also shown in Pictorial 9-8.

A byte of serial data written onto or read from a sector is defined as eight consecutive bit cells. See Pictorial 9-9 (Illustration Booklet, Page 13). The most significant bit cell is defined as bit cell 0 and the least significant as bit cell 7. When a specific data bit is referred to (such as bit 3), it is with respect to the corresponding bit cell (bit cell 3).



PICTORIAL 9-8

During a write operation, bit cell 0 of each byte is transferred to the disk drive first; while bit cell 7 is transferred last. Whenever data is read back from the drive, bit cell 0 of each byte is transferred first and bit cell 7 last. Pictorial 9-10 (Illustration Booklet, Page 13) shows the information written on a sector.

USART U802 converts the serial data to parallel data and parallel data to serial data. However, the serial data format required by U802 is NRZ. Thus, it is

necessary to convert the NRZ data to (to write) and from (to read) the bit cell format required by the floppy drive. This is accomplished by U812A, C, D; U813; and U816 for a write and U808; U815; U806D; U811A, B, C; and U812B for the read.

For the write operation, U816 divides the 2.048 MHz clock by 16. Its binary outputs are used by U813 which decodes the binary information into a 1-of-8 output. Each output is 0.977 μs in duration. (See Pictorial 9-11 Illustration Booklet, Page 13.) Output Q2 (pin 3) of U813 is the bit cell clock pulse and output Q6 (pin 7) is the data pulse. The QD output (pin 11) of U816 clocks the NRZ data from USART U802 and occurs every 7.678 μs. This defines the bit cell. The NRZ data is therefore, synchronized with the 2.048 MHz system clock, and, consequently, the bit cell clock.

The data bit pulse is AND'ed with the NRZ data by U812A to produce a pulse (pin 3 of U812A) if the NRZ is a '1' and no pulse if the NRZ data is a '0'. This information is OR'ed with the bit cell clock by U812D to produce the composite bit cell data pattern required by the floppy drive. U804 gates the write data with the write gate in order to prevent any extraneous writing to the diskette unless the write gate is on.

The read data signal is conditioned by U806C and U808 to produce a very short (50 ns) active high pulse which occurs at the leading edge of both the bit cell data and clock pulses coming from the drive. U815 is a divide-by-12 counter that separates the data from the clock pulses.

Assume as a starting point that U815 has counted to 12 and the ENT and ENP inputs (pins 7 and 10) are low, preventing any further counting. At this point, the output of U806D is high. This allows the next pulse from the drive to reset U815 by U811D. Since all data strings start with a number of consecutive zero bytes (no data pulse), the counter will synchronize on the clock bit and not on the data bits. During the time between reset and count 12 (5.859 μs), data pulses are accepted. Therefore, the pulse is assumed to be a clock pulse.

When a valid data pulse is received, the R-S flip-flop (U811A and U811B) is set (pin 6 goes high). This signifies that the data bit was a '1'. If no pulse was received within 5.859 μs after a clock pulse, then the R-S flip-flop is not set and the data bit was a '0'. At this time, a 12 count has been reached and the data is toggled into the USART (U802). The next pulse from the drive is a clock pulse which resets counter U815 and the R-S flip-flop; thereby starting a new cycle.

U802, in addition to converting serial bytes to parallel during a read operation and parallel bytes to serial during a write operation, also searches for a synchronizing byte at the beginning of each data and address block on each sector. A read from port 176 resets U802 to the search character mode and forces a '0' on the output (pin 3) of R-S flip-flop U809A and U809B. The USART now compares each byte as it is input to the special character in its sync character holding register. When a match is made, the USART receiver returns to the byte mode and R-S flip-flop U809A and U809B is set to a '1'. This informs the processor that the proper match was made.

Port 177 reads the floppy disk status and sync character match and controls the floppy disk with respect to its motor, track position, and write gate. Port 176 resets the USART to the search character mode and writes the search character to the USART. Port 175 reads and sets the USART status. Port 174 receives and transmits the data characters to the USART. The decoding for these various ports is accomplished by U814A, U814B, and U801. U803 latches the floppy control signals and U805 latches the floppy status signals.

REPLACEMENT PARTS LIST

POWER SUPPLY

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
CAPACITORS, Electrolytic		
C101	25-902	10,000 μF
C102	25-906	4700 μF
C103	25-902	10,000 μF
C104	25-891	470 μF
DIODES		
BR1	56-67	Bridge rectifier
D101	57-42	3A1
Diodes (cont'd.)		
D102	57-42	3A1
D103	57-42	3A1
D104	57-42	3A1
D109	57-65	1N4002
D110	57-65	1N4002
D111	57-65	1N4002
D112	57-65	1N4002
INTEGRATED CIRCUIT		
See "Semiconductor Identification."		

VIDEO CIRCUIT BOARD

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
RESISTORS		
NOTE: The following resistors are 5%, 1/2-watt unless otherwise specified.		
R201	6-105	1 MΩ
R202	6-472	4700 Ω
R203	6-102	1000 Ω
R204	6-682	6800 Ω
R205	6-472	4700 Ω
R206	6-101	100 Ω
R207	6-6491	6490 Ω, 1%
R208	6-1871-12	1870 Ω, 1/4-watt, 1%
R209	6-102	1000 Ω
R210	Not used	
R211	3-6-2	.51 Ω, 2-watt
Resistors (cont'd.)		
R212	3-57-5	1500 Ω, 5-watt, 10%
R213	6-470	47 Ω
R214	6-102	1000 Ω
R215	1-50-2	820 Ω, 2-watt
R216	10-918	500 control
R217	6-750	75 Ω
R218	6-330	33 Ω
R219	6-220	22 Ω
R220	Not used	
R221	6-471	470 Ω
R222	6-152	1500 Ω
R223	6-332	3300 Ω
R224	6-104	100 k
R225	6-223	22 kΩ
R226	10-390	20 kΩ control

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
Resistors (cont'd.)		
R227	6-224	220 kΩ
R228	10-390	20 kΩ control
R229	6-273	27 kΩ
R230	Not used	
R231	6-822	8200 Ω
R232	6-103	10 kΩ
R233	6-623	62 kΩ
R234	6-473	47 kΩ
R235	6-103	10 kΩ
R236	6-122	1200 Ω
R237	6-101	100 Ω
R238	6-105	1 MΩ
R239	6-103	10 kΩ
R240	Not used	
R241	6-182	1800 Ω
R242	6-223	22 kΩ
R243	6-222	2200 Ω
R244	6-150	15 Ω
R245	6-471	470 Ω
R246	6-279	2.7 Ω
R247	6-279	2.7 Ω
R248	6-479	4.7 Ω
R249	6-102	1000 Ω
R250	Not used	
R251	6-223	22 kΩ
R252	6-273	27 kΩ
R253	10-311	5000 Ω control
R254	6-822	8200 Ω
R255	6-273	27 kΩ
R256	6-392	3900 Ω
R257	6-201	200 Ω
R258	6-470	47 Ω
R259	6-392	3900 Ω
R260	Not used	
R261	Not used	
R262	6-101	100 Ω
R263	3-22-2	1.2 Ω, 2-watt, 10%
R264	3-22-2	1.2 Ω, 2-watt, 10%
R265	6-104	100 kΩ
R266	6-331	330 Ω
R267	6-473	47 kΩ
R268	10-941	100 kΩ control
R269	6-823	82 kΩ
R270	Not used	
R271	10-1049	2 MΩ
R272	6-394	390 Ω
R273	6-102	1000 Ω
R274	6-472	4700 Ω
R275	6-335	3.3 MΩ
R276	6-335	3.3 MΩ

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
CAPACITORS		
C201	21-140	.001 μF ceramic
C202	25-865	10 μF electrolytic
C203	21-176	.01 μF ceramic
C204	25-220	10 μF tantalum
C205	25-220	10 μF tantalum
C206	21-176	.01 μF ceramic
C207	20-106	390 pF Mica
C208	21-176	.01 μF ceramic
C209	25-883	47 μF electrolytic
C210	Not used	
C211	27-145	.22 μF Mylar
C212	27-145	.22 μF Mylar
C213	25-841	4.7 μF tantalum
C214	29-32	6800 pF polystyrene
C215	25-865	10 μF electrolytic
C216	21-121	56 pF ceramic
C217	25-890	330 μF electrolytic
C218	21-75	100 pF ceramic
C219	21-176	.01 μF ceramic
C220	Not used	
C221	29-22	4700 pF polystyrene
C222	21-75	100 pF ceramic
C223	21-176	.01 μF ceramic
C224	29-22	4700 pF polystyrene
C225	27-73	.047 μF Mylar
C226	25-882	22 μF electrolytic
C227	25-220	10 μF tantalum
C228	25-882	22 μF electrolytic
C229	29-56	.006 μF polypropylene
C230	Not used	
C231	25-299	1.5 μF electrolytic
C232	29-57	.22 μF polypropylene
C233	27-206	1 μF polycarbonate
C234	21-193	.005 μF spark gap
C235	25-882	22 μF electrolytic
C236	21-122	.02 μF ceramic
C237	21-122	.02 μF ceramic
C238	21-122	.02 μF ceramic
C239	25-911	22 μF electrolytic
DIODES — TRANSISTORS — IC's		
See "Semiconductor Identification."		
INDUCTORS — CHOKES — TRANSFORMERS		
L201	45-39	4.65 μH choke
L202	40-581	620 μH inductor
L203	45-42	8.75 μH choke
L204	40-1947	19 μH inductor
L205	40-1948	52 μH inductor
T201	51-197	Driver transformer
T202	51-196	Flyback transformer

TERMINAL LOGIC CIRCUIT BOARD

RESISTORS

NOTE: All resistors are 1/4-watt, 5%.

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
R401	6-222-12	2200 Ω
R402	6-102-12	1000 Ω
R403	6-222-12	2200 Ω
R404	6-561-12	560 Ω
R405	6-561-12	560 Ω
R406	6-102-12	1000 Ω
R407	6-103-12	10 kΩ
R408	6-102-12	1000 Ω
R409	6-102-12	1000 Ω
R410	6-331-12	330 Ω
R411	6-224-12	220 kΩ
R412	6-224-12	220 kΩ
R413	6-103-12	10 kΩ
R414	6-472-12	4700 Ω
R415-R445	6-103-12	10 kΩ
R466	6-102-12	1000 Ω
RP1	9-98	220 kΩ resistor network
RP2	9-98	220 kΩ resistor control

CAPACITORS

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
C401-C404	25-221	2.2 μF tantalum
C405	25-276	4.7 μF tantalum
C406	25-221	2.2 μF tantalum
C407	25-276	4.7 μF tantalum
C408	25-221	2.2 μF tantalum
C409	25-221	2.2 μF tantalum
C410	Not used	
C411	20-101	47 pF Mica
C412	21-176	.01 μF ceramic

Capacitors (cont'd.)

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
C413	20-103	150 pF Mica
C414	25-223	47 μF tantalum
C415	25-221	2.2 μF tantalum
C416	21-192	.1 μF, 50 V ceramic
C417	25-221	2.2 μF tantalum
C418	25-220	10 μF tantalum
C419	21-46	.005 μF ceramic
C420	Not used	
C421	25-223	47 μF tantalum
C422-C440	21-95	.1 μF, 10 V ceramic
C441	21-143	.05 μF ceramic
C442	21-143	.05 μF ceramic
C443	21-143	.03 μF ceramic
C444	21-143	.05 μF ceramic
C445	21-143	.05 μF ceramic
C446-C474	21-85	.1 μF, 10 V ceramic
C475	21-140	.001 μF ceramic

MISCELLANEOUS

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
S401	60-621	Dip Switch
S402	60-621	Dip switch
Y401	404-613	12.288 MHz crystal

DIODES — TRANSISTOR — IC's

See "Semiconductor Identification."

CPU LOGIC CIRCUIT BOARD

RESISTORS

NOTE: All resistors are 1/4-watt, 5%.

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
R501	6-561-12	560 Ω
R502	6-561-12	560 Ω
R503	6-102-12	1000 Ω
R504	6-102-12	1000 Ω
R505	Not used	
R506	6-331-12	330 Ω
R507	6-471-12	470 Ω

Resistors (cont'd.)

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
R508	6-103-12	10 kΩ
R509	6-393-12	39 kΩ
R510	6-103-12	10 kΩ
R511	6-102-12	1000 Ω
R512	6-151-12	150 Ω
R513	6-151-12	150 Ω
R514	6-472-12	4700 Ω
R515	6-152-12	1500 Ω
R516	6-105-12	1 MΩ

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
CAPACITORS		
C501	20-103	150 pF mica
C502	21-185	.01 μF ceramic
C503	20-101	47 pF mica
C504	20-106	390 pF mica
C505	20-171	820 pF mica
C506	25-282	68 μF tantalum
C507	25-221	2.2 μF tantalum
C508	25-221	2.2 μF tantalum
C509	Not used	
C510	Not used	
C511	25-221	2.2 μF tantalum
C512	25-221	2.2 μF tantalum
C513	25-221	2.2 μF tantalum
C514	25-221	2.2 μF tantalum
C515	25-221	2.2 μF tantalum
C516	25-221	2.2 μF tantalum
C517	25-221	2.2 μF tantalum

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
Capacitors (cont'd.)		
C518	20-114	270 pF mica
C519	20-114	270 pF
C520	25-221	2.2 μF tantalum
C521	20-77	24 pF mica
C522	21-147	47 pF ceramic
C523	21-3	10 pF ceramic
C524-C593	21-185	.01 μF ceramic
MISCELLANEOUS		
S501	60-621	8-section switch
X501	404-613	12.288 MHz crystal
X502	404-608	1.843 MHz crystal
DIODE — TRANSISTOR — IC's		
See "Semiconductor Identification."		

CHASSIS PARTS

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
C1	25-857	1500 μF electrolytic capacitor
T1	54-969	Power transformer
T2	58-19	Yoke
SW1	60-54	120/240 switch
SW2	60-608	NOR/LOW switch
SW3	60-619	Power switch
F1	421-23	1-ampere fuse
F1	421-25	1.5-ampere fuse
	401-163	Speaker
	411-838	CRT

FLOPPY DISK INTERFACE CIRCUIT BOARD

RESISTORS

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
R801	6-151-12	150 Ω
R802	6-151-12	150 Ω
R803	6-151-12	150 Ω
R804	6-151-12	150 Ω
R805	6-151-12	150 Ω

CAPACITORS

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
C801	21-185	.01 μF ceramic
C802	21-185	.01 μF ceramic
C803	21-185	.01 μF ceramic
C804	21-185	.01 μF ceramic
C805	21-185	.01 μF ceramic
C806	21-164	.0015 μF ceramic
C807	21-185	.01 μF ceramic
C808	21-185	.01 μF ceramic
C809	21-185	.01 μF ceramic
C810	21-185	.01 μF ceramic
C811	20-99	22 pF mica

SERIAL INTERFACE CIRCUIT BOARD

RESISTOR

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
R601	6-102-12	1000 Ω, 1/4-watt, 5%

CAPACITORS

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
C601	25-221	2.2 μF tantalum
C602	25-221	2.2 μF tantalum
C603	25-221	2.2 μF tantalum
C604	25-221	2.2 μF tantalum

INTEGRATED CIRCUITS

See "Semiconductor Identification."

CASSETTE INTERFACE CIRCUIT BOARD

RESISTORS

(All resistors are 1/4-watt, 5% unless marked otherwise.)

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
R701	6-391-12	390 Ω
R702	6-391-12	390 Ω
R703	6-332-12	3300 Ω
R704	6-332-12	3300 Ω
R705	6-104-12	100 kΩ
R706	6-103-12	10 kΩ
R707	6-220-12	22 Ω
R708	6-101-12	100 Ω
R709	6-103-12	10 kΩ
R711	6-104-12	100 kΩ
R712	6-332-12	3300 Ω
R713	6-332-12	3300 Ω
R714	6-332-12	3300 Ω
R715	6-104-12	100 kΩ
R716	6-332-12	3300 Ω
R717	6-9762-12	97.7 kΩ, 1%
R718	6-102-12	1000 Ω
R719	6-1583-12	158 k, 1%
R721	6-391-12	390 Ω
R722	6-332-12	3300 Ω
R723	6-104-12	100 kΩ
R724	6-103-12	10 kΩ
R725	6-104-12	100 kΩ
R726	6-101-12	100 Ω
R727	6-332-12	3300 Ω
R728	6-332-12	3300 Ω
R729	6-103-12	10 kΩ
R731	6-103-12	10 kΩ
R732	6-104-12	100 kΩ
R733	6-332-12	3300 Ω
R734	6-279	2.7 Ω, 1/2-watt
R735	6-332-12	3300 Ω
R736	6-279	2.7 Ω, 1/2-watt

CAPACITORS

CIRCUIT Comp. No.	HEATH Part No.	DESCRIPTION
C701	25-276	4.7 μF tantalum
C702	25-276	4.7 μF tantalum
C703	25-276	4.7 μF tantalum
C704	21-185	.01 μF ceramic
C705	21-185	.01 μF ceramic
C706	21-56	470 pF ceramic
C707	21-25	1300 pF ceramic
C708	21-185	.01 μF ceramic
C709	21-185	.01 μF ceramic
C711	21-185	.01 μF ceramic
C712	21-185	.01 μF ceramic
C713	21-185	.01 μF ceramic
C714	21-185	.01 μF ceramic
C715	27-147	.0056 Mylar
C716	27-129	.047 μF Mylar
C717	25-276	4.7 μF tantalum
C718	29-39	3000 pF polystyrene
C719	27-147	.0056 μF Mylar
C721	29-39	3000 pF polystyrene
C722	21-185	.01 μF ceramic
C723	21-185	.01 μF ceramic
C724	21-185	.01 μF ceramic
C725	21-185	.01 μF ceramic
C726	21-185	.01 μF cermiac
C727	25-223	
C729	21-185	.01 μF ceramic
C731	21-185	.01 μF ceramic
C732	21-185	.01 μF ceramic

RELAYS

K701	40-1667	Reed relay
K702	40-1667	Reed relay

DIODES

See "Semiconductor Identification."

SEMICONDUCTOR IDENTIFICATION

This section is divided into two parts; "Component Number Index" and "Part Number Index." The first section provides a cross-reference between semiconductor component numbers and their respective Part Numbers. The component numbers are listed in numerical order. The second section provides a lead configuration detail (basing diagram) for each semiconductor Part Number. The Part Numbers in the second section are also listed in numerical order.

COMPONENT NUMBER INDEX

This index shows the Part Number of each semiconductor in the Computer.

POWER SUPPLY CIRCUIT BOARD

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
BR1	57-67
D101	57-42
D102	57-42
D103	57-42
D104	57-42
D109	57-65
D110	57-65
D111	57-65
D112	57-65
U101	442-30

VIDEO CIRCUIT BOARD

Diodes

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
D201	56-56
D202	56-73
D203	56-94
D204	56-56
D205	56-58
D206	56-93
D207	56-56
D208	57-614
D209	57-27
D211	57-64
D212	57-27
D213	56-56

Transistors

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
Q201	417-823
Q202	417-885
Q203	417-822
Q204	417-821
Q205	417-926
Q206	417-926
Q207	417-923
Q208	417-924
Q209	417-811
Q211	417-924
Q212	417-874
Q213	417-282
Q214	417-834
Q215	417-875
Q216	417-195
Q217	417-923
Q218	417-927

Integrated Circuits

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
U201	442-53
U202	442-53

TERMINAL LOGIC CIRCUIT BOARD

Diodes

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
D401	56-56
D402	56-56

Transistor

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
Q401	417-937
Q402	417-937

Integrated Circuits

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
U401	442-54
U402	442-54
U403	442-663
U404	442-664
U405	442-630
U411	443-18
U412	443-757
U413	443-34
U414	443-760
U415	443-792
U416	443-728
U417	443-727
U418	443-877
U419	443-877
U421	443-881
U422	444-46

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
U424	443-721
U425	443-721
U426	443-733
U427	443-779
U428	443-730
U429	443-875
U430	443-37
U431	443-913
U432	443-730
U433	443-791
U434	443-791
U435	443-18
U436	443-792
U437	443-791
U451	443-874
U452	443-794

Integrated Circuits (Cont'd)

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
U453	443-795
U461	443-906
U462	443-730
U463	443-799
U464	443-799
U465	443-799
U466	443-728
U467	443-764
U468	443-764
U469	443-764
U470	443-764
U471	443-885
U472	443-805
U473	444-29
U474	443-892
U475	443-915
U476	443-805
U477	443-780
U478	443-900
U479	443-891

Resistor Packs

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
RP1	9-98
RP2	9-98

CPU LOGIC CIRCUIT BOARD

Diode

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
D501	56-56

Transistor

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
Q501	417-821

Integrated Circuits

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
U501	443-18
U502	443-34
U503	443-760
U504	443-881
U506	443-730
U507	443-727
U508	443-792
U509	443-824
U510	443-837
U511	443-837
U512	443-730
U513	443-824
U514	443-824
U515	443-779
U516	444-41
U517	444-42
U518	444-40
U519	Not Used
U520	444-19
U521	443-754
U522	Not Used
U523	443-764
U524	Not Used
U525	443-764
U526-U4549	443-904
U550	444-43
U551	443-791
U552	443-805
U553	443-875
U554	443-732
U555	443-730
U556	443-730
U557	443-912
U558	443-754
U559	443-794
U560	443-795
U561	443-874
U562	443-792
U565	442-664
U566	442-665
U567	442-663
U568	442-663
U569	442-665

FLOPPY DISK INTERFACE CIRCUIT BOARD

Integrated Circuits

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
U102	442-30
U103	442-650
U801	443-822
U802	443-856
U803	443-805
U804	443-77
U805	443-857
U806	443-872
U807	443-73
U808	443-22
U809	443-728
U811	443-792
U812	443-728
U813	443-807
U814	443-875
U815	443-757
U816	443-757

SERIAL INTERFACE CIRCUIT BOARD

Integrated Circuits

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
U601	443-818
U602	443-874
U604	443-874
U605	443-795
U606	443-794
U609	443-795
U611	443-794

CASSETTE INTERFACE CIRCUIT BOARD

Diodes

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
D701	56-56
D702	56-59
D703	56-56
D704	56-56
LED701	412-640

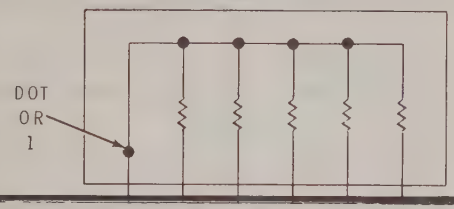
Integrated Circuits

CIRCUIT COMPONENT NUMBER	HEATH PART NUMBER
U701	443-797
U702	443-875
U703	443-811
U704	443-730
U705	443-811
U706	443-792
U707	443-776
U708	443-755
U709, U710	442-53
U711	442-616
U712	443-73
U713	443-34
U714	443-760
U715	443-728

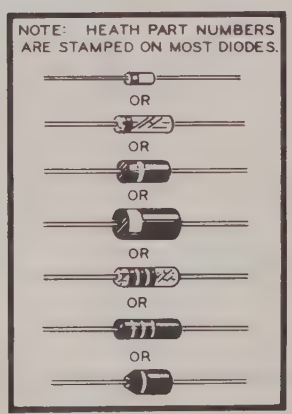
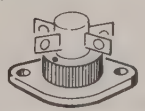
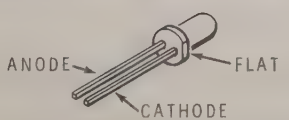
PART NUMBER INDEX

This index shows a lead configuration detail (basing diagram) of each semiconductor part number.

RESISTOR PACK

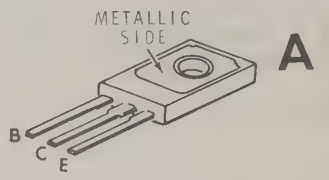
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
9-98		220 kΩ resistor network	

DIODES

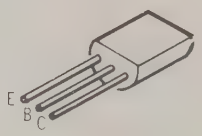
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
56-56	1N4149	10 mA, 75 V	
56-58	1N709A	Zener, 6.2 V, 25 mA	
56-73	MZ2360	Compensation	
56-93	FD333	225 mA, 125 V	
56-94		Zener, 12.8 V, 12 mA	
57-27	1N2071	SI Rect 1 A, 600 V	
57-42	3A1	SI rect 3A, 100 V	
57-64	DRS-110	SI Rect 1A, 1000 V	
57-65	1N4002	SI Rect 1A, 100 V	
57-614	MR-508	SI Rect 3A, 800 V	
57-67	10A20	Diode Bridge	
412-640		Light-emitting Diode	

TRANSISTORS

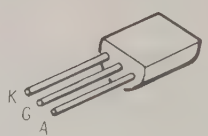
HEATH PART NUMBER	MAY BE REPLACED WITH	BASING DIAGRAM
417-195	MJE340	A
417-282	MJ2841	E
417-811	MPSL01	B
417-821	MPSA06	B
417-822	MPSA56	B
417-823	MPU131	C
417-834	MPSU10	D
417-874	2N3906	B
417-875	2N3904	B
417-885	MPSA65	B
417-923	BU500	E
417-924	MJE172	A
417-926	MPSU06	D
417-927	MPSA93	B
417-932	MJE182	A
417-937	MPS2369	B



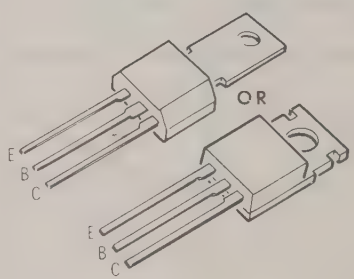
A



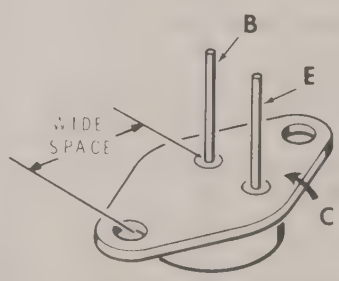
B



C

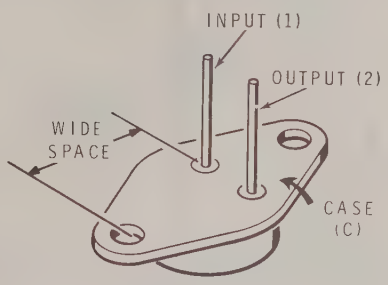
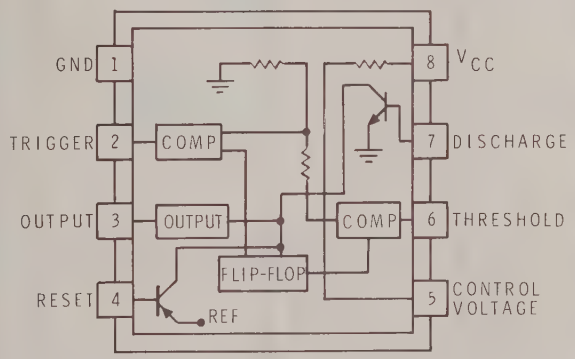
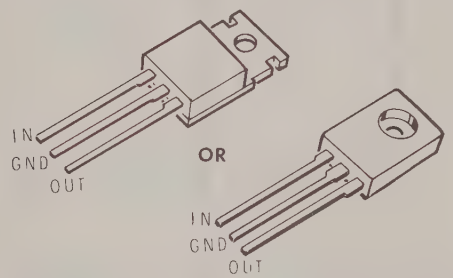
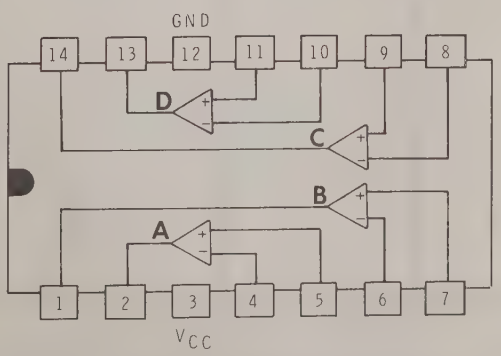


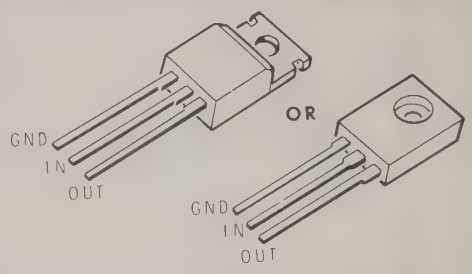
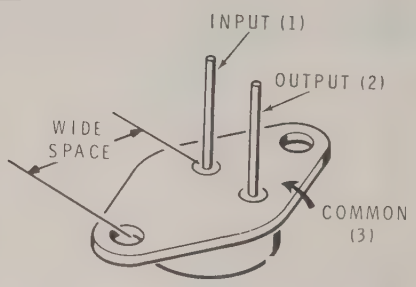
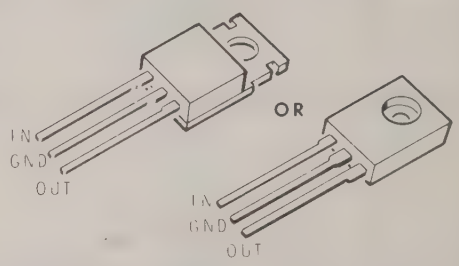
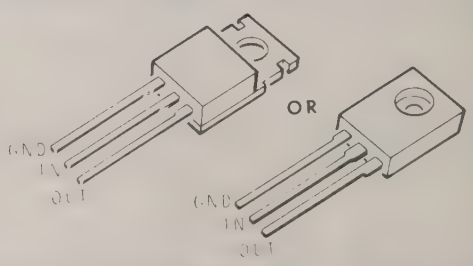
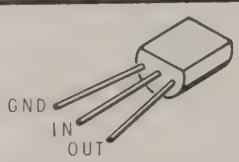
D

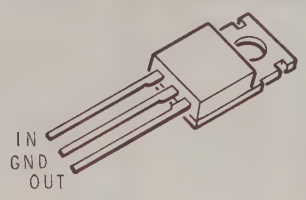
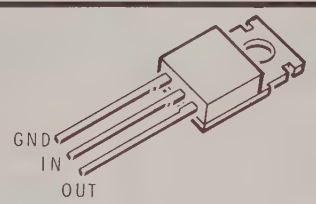
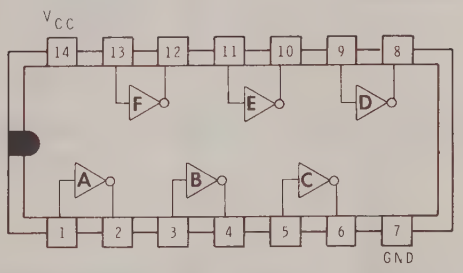
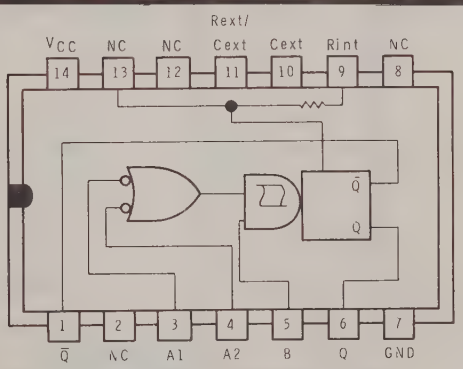
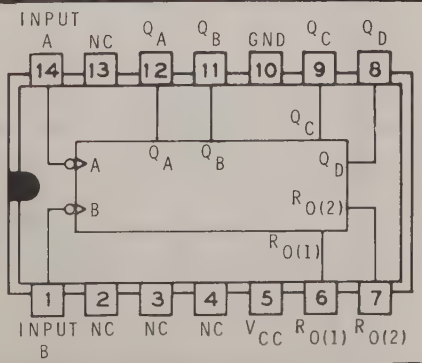


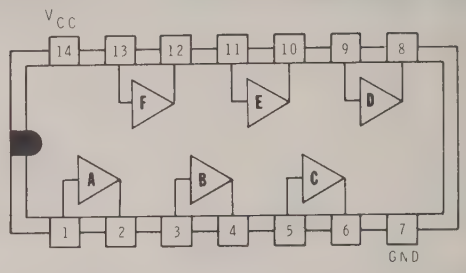
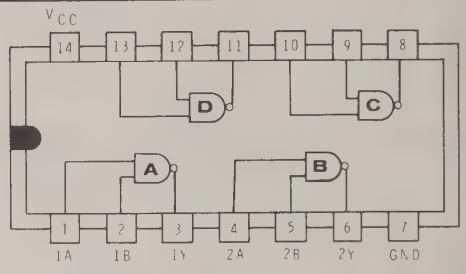
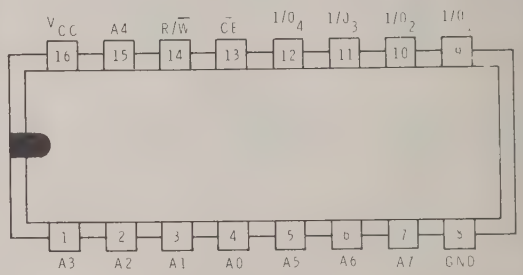
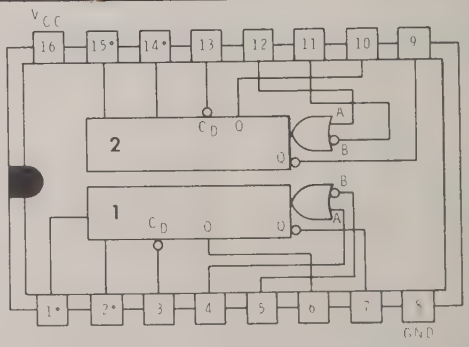
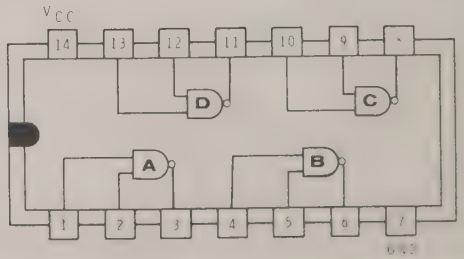
E

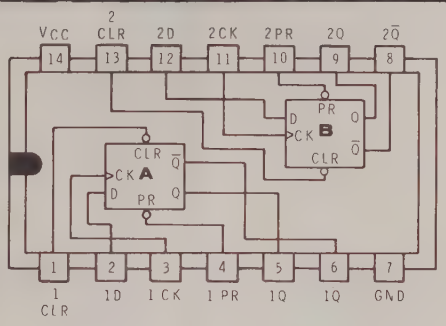
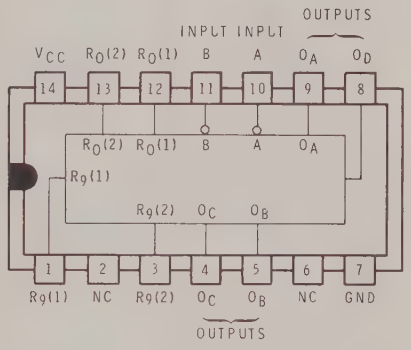
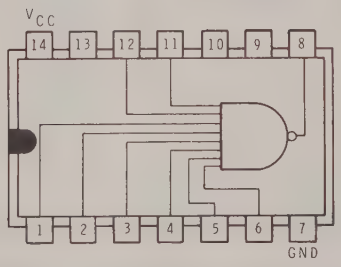
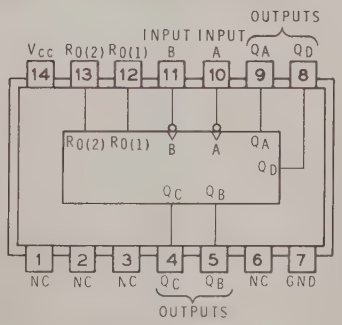
INTEGRATED CIRCUITS

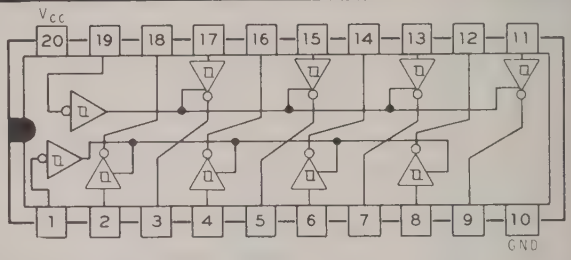
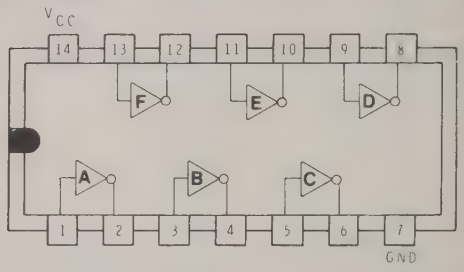
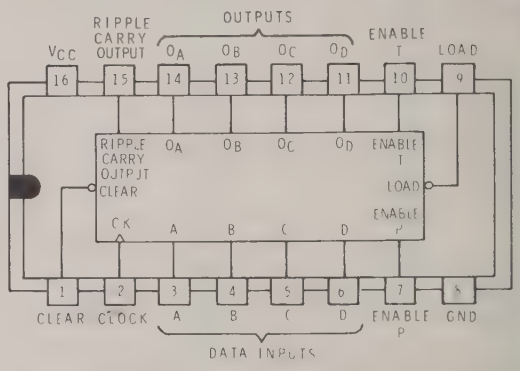
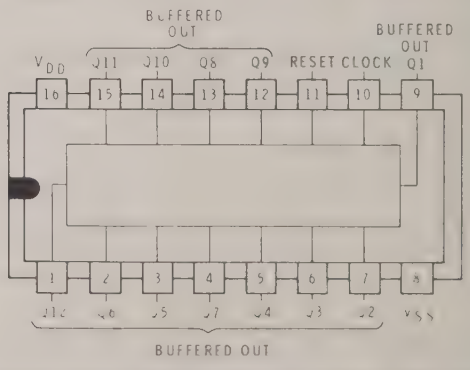
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
442-30	UA309K LM309	5-volt Regulator	
442-53	555	Timer	
442-54	7805	+5 V Regulator	
442-616	LM3302N, LM2901, or μA775 (selected)	Quad Operational Amplifier	

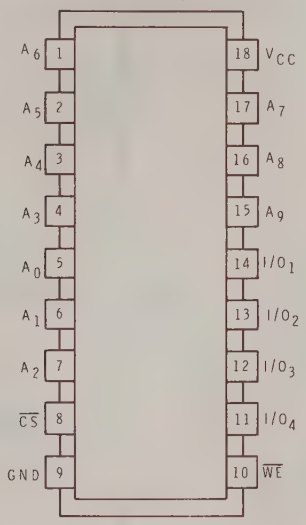
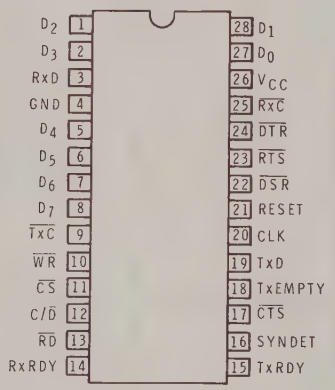
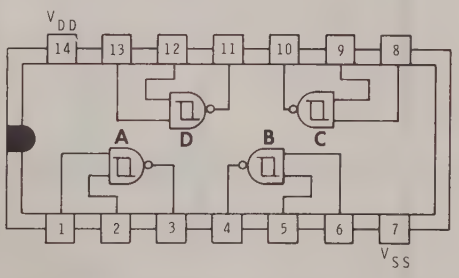
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
442-630	7905.2	−5.2 V Regulator	
442-650	78H12	+12 V. 5A Regulator	
442-663	78M12CKC	+12 V Regulator	
442-664	79M12CKC	−12 V Regulator	
442-665	79L05AC	−5 V Regulator	

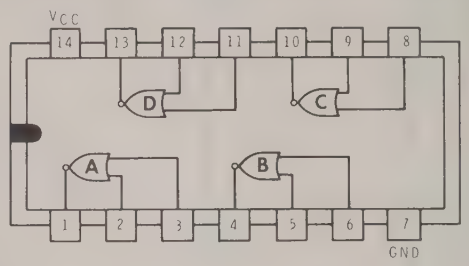
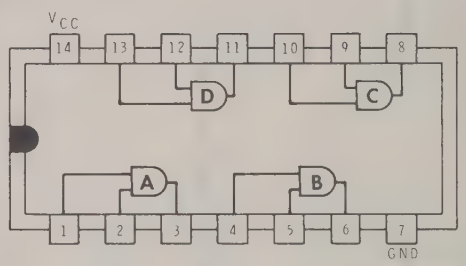
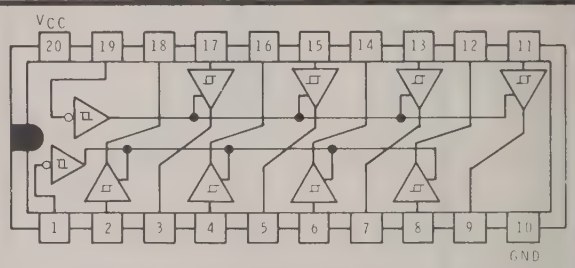
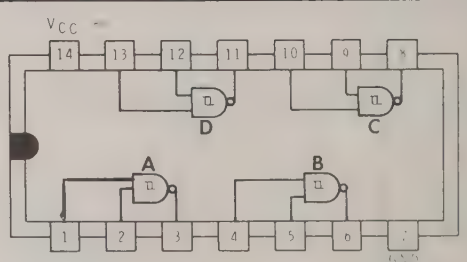
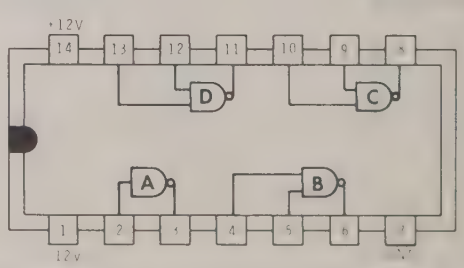
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
442-674	UA7812	+12 V Regulator	
442-683	79M05	−5 V Regulator	
443-18	7404	Hex Inverter	
443-22	74121	Monostable Multivibrator	
443-34	7492	Divide-By-Twelve Counter	

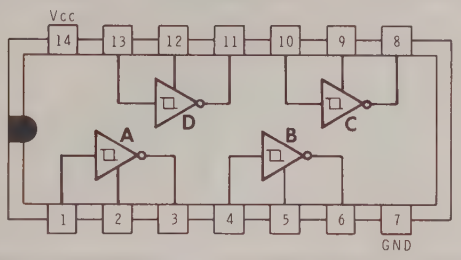
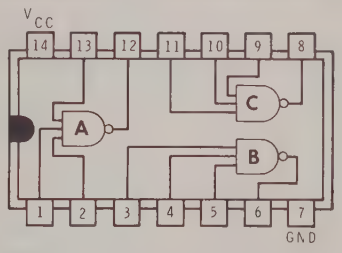
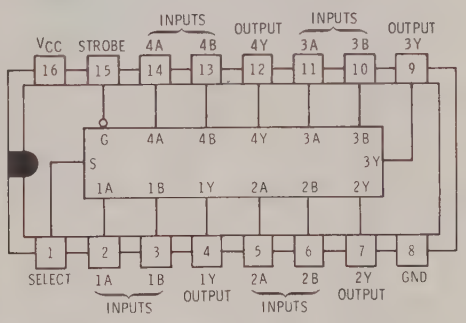
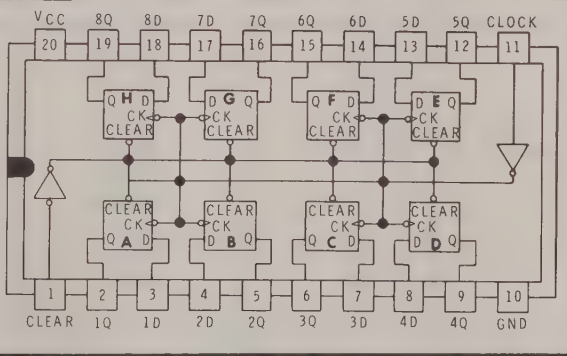
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-73	7416	Line Driver	
443-77	7438	Quadruple 2-Input Positive-NAND Buffers With Open-collector Outputs	
443-721	2112-2	256 × 4 RAM	
443-727	96L02	Dual Monostable	
443-728	74LS00	Quad 2-input NAND	

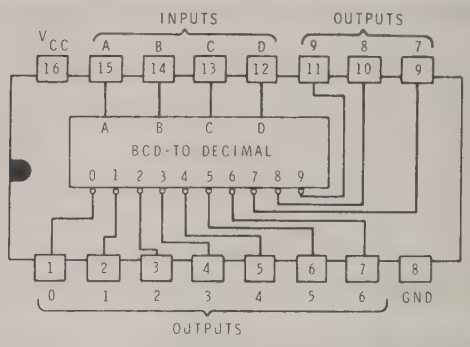
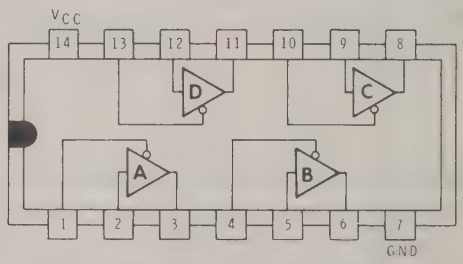
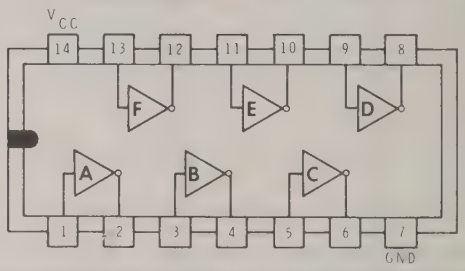
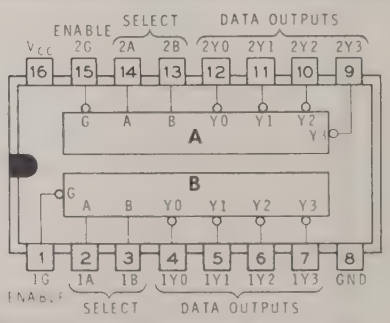
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-730	74LS74	Dual D Flip-flop	
443-731	SN74LS290	BCD Counter	
443-732	SN74LS30	8-input NAND	
443-733	74LS293	4-Bit Binary Counter	

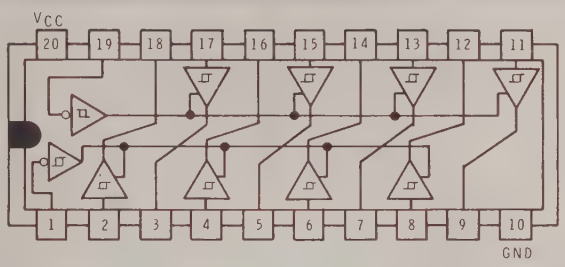
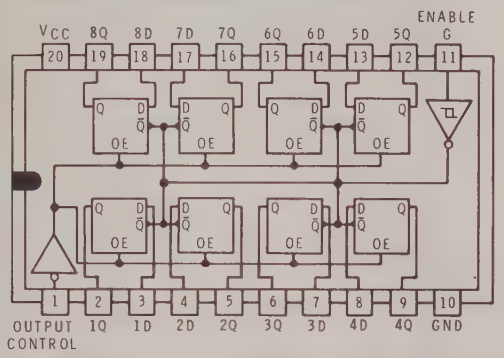
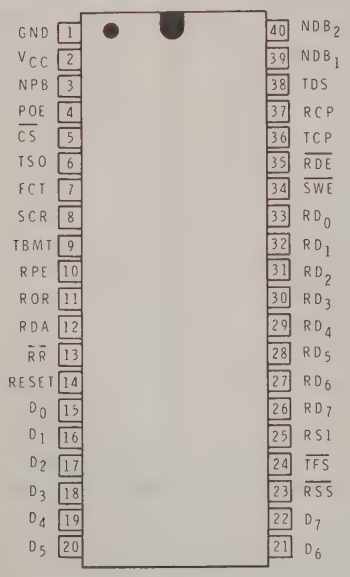
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-754	74LS240	Octal buffer, 3-state outputs	
443-755	74LS04	Hex Buffer	
443-757	74LS161	4-Bit Binary Counter	 RIPPLE CARRY OUTPUT OUTPUTS ENABLE T LOAD Vcc 16 15 14 13 12 11 10 9 RIPPLE CARRY OUTPUT CLEAR CK A B C D ENABLE P CLEAR CLOCK DATA INPUTS GND
443-760	4040	12-Bit Binary	

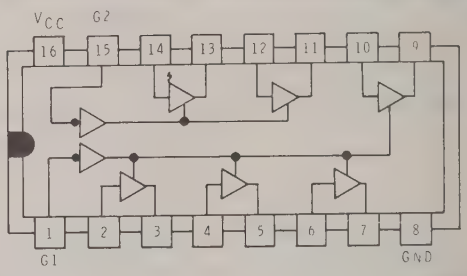
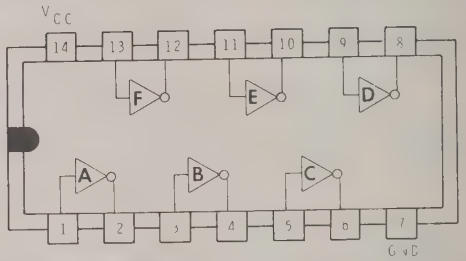
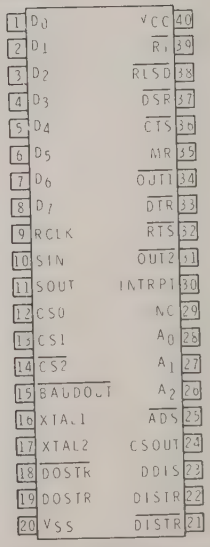
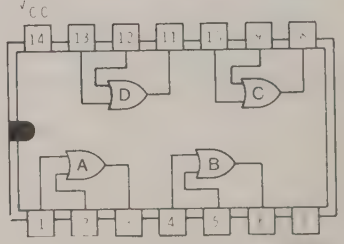
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-764	2114	1k × 4 RAM	
443-776	8251	USART	
443-778	4093	Quad 2-input NAND SCHMITT Trigger	

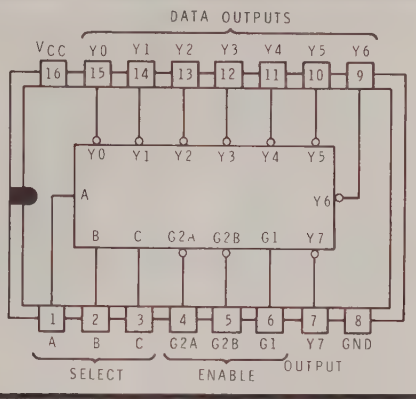
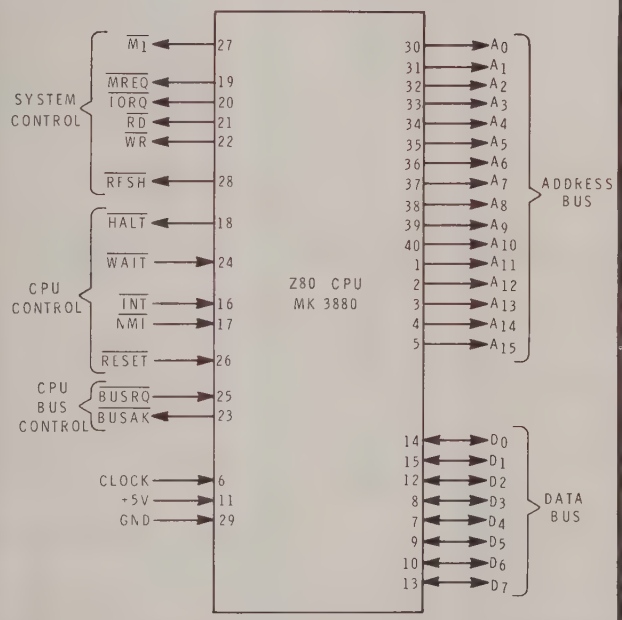
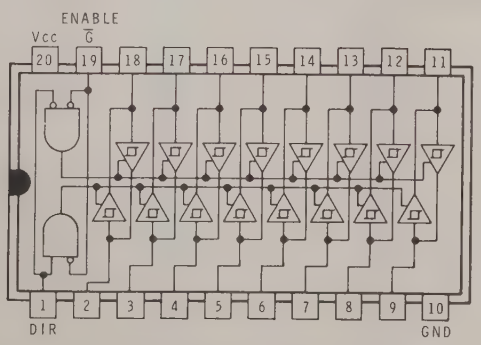
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-779	74LS02	Quad 2-input Positive-NOR gates	
443-780	74LS08	Quad 2-input Positive-AND Gates	
443-791	74LS244	Noninverting 3-state output octal buffers	
443-792	74LS132	Quad 2-input Positive-NAND Schmitt Triggers	
443-794	75188 or 1488	EIA Driver	

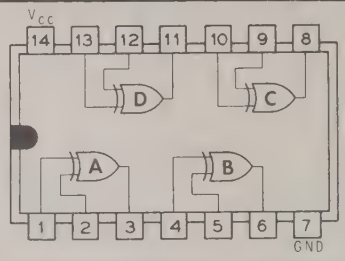
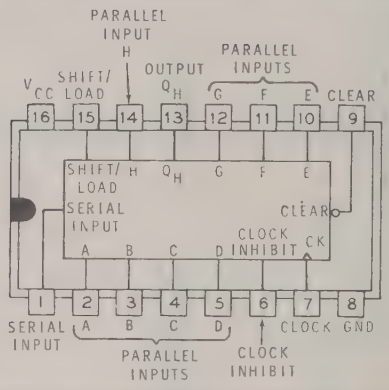
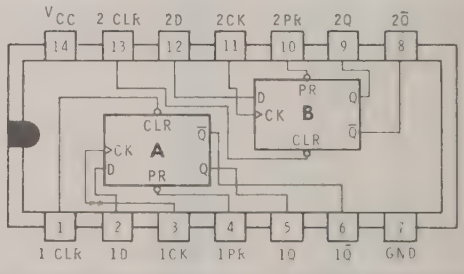
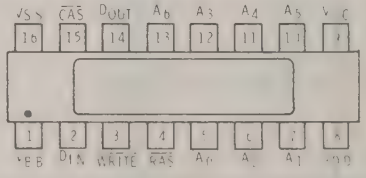
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-795	75189 or 1489	EIA Receiver	
443-797	74LS10	Triple 3-input Positive-NAND Gate	
443-799	74LS157	Quad 2-line-to-1-line Multipliers	 INPUTS OUTPUT INPUTS OUTPUT VCC STROBE 4A 4B 4Y 3A 3B 3Y 16 15 14 13 12 11 10 9 G 4A 4B 4Y 3A 3B S 3Y 1A 1B 1Y 2A 2B 2Y 1 2 3 4 5 6 7 8 SELECT 1A 1B 1Y 2A 2B 2Y GND INPUTS OUTPUT INPUTS OUTPUT
443-805	74LS273	Octal D Flip-flop with clear	

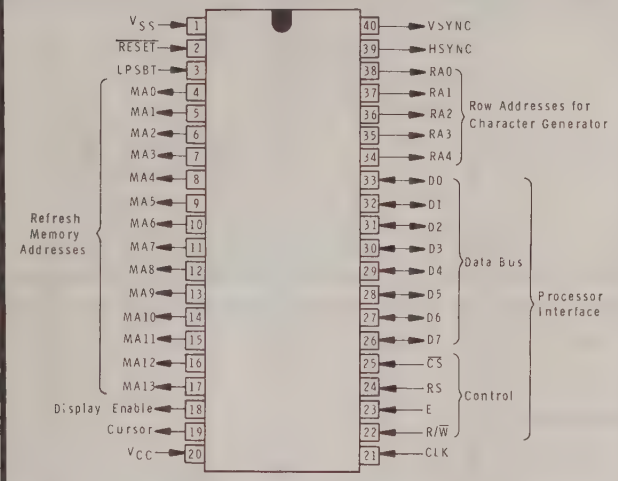
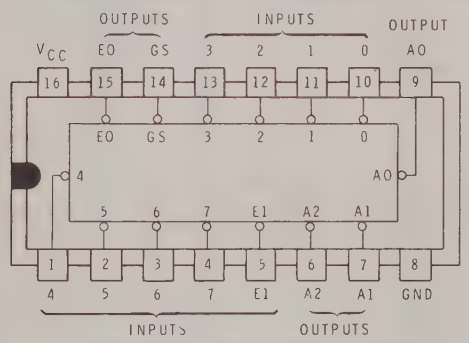
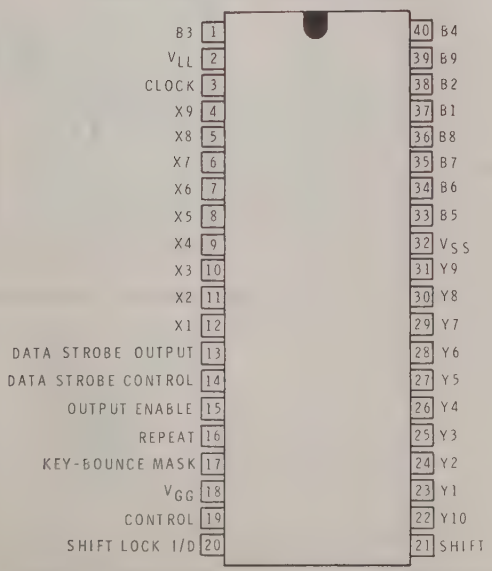
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-807	74LS42	BCD-to-Decimal Decoder	
443-811	74LS125	Quad bus buffer gate with 3-state outputs	
443-818	74LS05	Hex Inverter	
443-822	74LS139	Dual 2-to-4-line decoder	 ENABLE SELECT DATA OUTPUTS VCC 2G 2A 2B 2Y0 2Y1 2Y2 2Y3 16 15 14 13 12 11 10 9 G A B Y0 Y1 Y2 A B 1 2 3 4 5 6 7 8 1G 1A 1B 1Y0 1Y1 1Y2 1Y3 GND SELECT DATA OUTPUTS

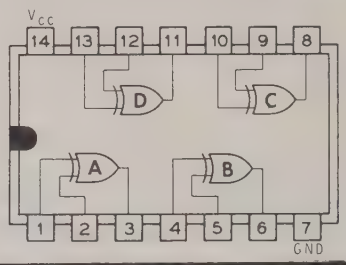
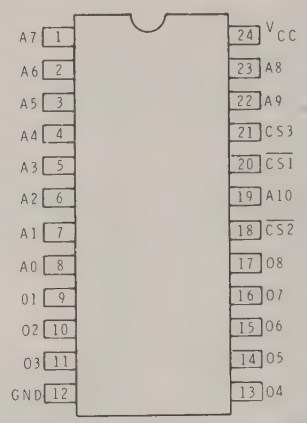
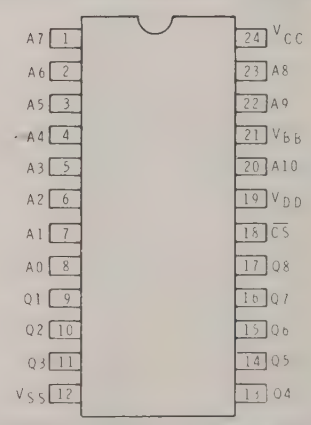
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-824	74LS241	Octal buffer	
443-837	74LS373	Octal D latch	
443-856	S2350	Universal Synchronous Receiver/ Transmitter (USRT)	 GND 1 VCC 2 NPB 3 POE 4 CS 5 TSO 6 FCT 7 SCR 8 TBMT 9 RPE 10 ROR 11 RDA 12 RR 13 RESET 14 D0 15 D1 16 D2 17 D3 18 D4 19 D5 20 40 NDB2 39 NDB1 38 TDS 37 RCP 36 TCP 35 RDE 34 SWE 33 RD0 32 RD1 31 RD2 30 RD3 29 RD4 28 RD5 27 RD6 26 RD7 25 RS1 24 TFS 23 RSS 22 D7 21 D6

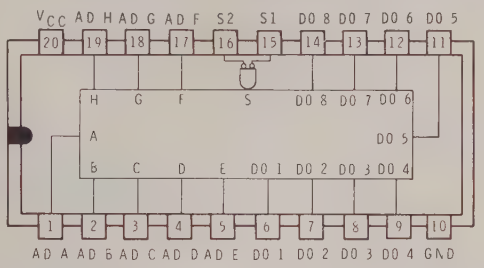
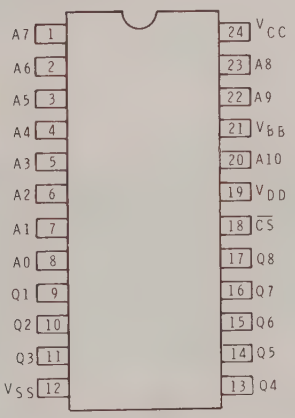
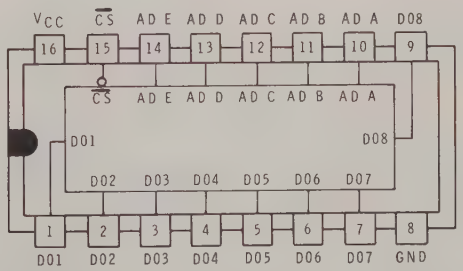
HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-857	74LS367	Hex Bus Drivers	
443-872	74LS14	Hex SCHMITT-Trigger Inverters	
443-874	8250	ACE	 1 D0 2 D1 3 D2 4 D3 5 D4 6 D5 7 D6 8 D7 9 RCLK 10 SIN 11 SOUT 12 CS0 13 CS1 14 CS2 15 BAUDOUT 16 XTAL1 17 XTAL2 18 DOSTR 19 DOSTR 20 VSS 40 VCC 39 R 38 RLSD 37 DSR 36 CTS 35 MR 34 OUT1 33 DTR 32 RTS 31 OUT2 30 INTRPT 29 NC 28 A0 27 A1 26 A2 25 ADS 24 CSOUT 23 DDIS 22 DISTR 21 DISTR
443-875	74LS32	Quad 2-input Positive OR Gates	

HEATH PART NUMBER	MAY BE REPALCED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-877	74LS138	3-to-8-line Decoder	
443-881	Z 80	Microprocessor	 SYSTEM CONTROL M1 MREQ IORQ RD WR RFSH CPU CONTROL HALT WAIT INT NMI RESET CPU BUS CONTROL BUSRQ BUSAK CLOCK +5V GND Z80 CPU MK 3880 ADDRESS BUS A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 DATA BUS D0 D1 D2 D3 D4 D5 D6 D7
443-885	74LS245	Octal bus transceiver	

HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-891	74LS86	Quad 2-input Exclusive — OR	
443-892	74LS166	8-bit shift register	 PARALLEL INPUT H SHIFT/ LOAD V_{CC} OUTPUT Q_H PARALLEL INPUTS G F E CLEAR 16 15 14 13 12 11 10 9 SHIFT/ LOAD SERIAL INPUT CLEAR CLOCK INHIBIT CK A B C D 1 2 3 4 5 6 7 8 SERIAL INPUT A B C D PARALLEL INPUTS CLOCK INHIBIT CLOCK GND
443-900	74S74	Dual-D Flip-flop	
443-904	MK4116-4	16k × 1 RAM	

HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-906	6845	CRT Controller	
443-912	74LS148	8-line-to-3-line Priority encoder	
443-913	S740	Keyboard Encoder	

HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
443-915	74S86	Quad 2-input Exclusive — OR	
444-19 444-29 444-37	2316 or 8316	2k × 8-bit ROM (Available only from Heath Co.)	
444-39 444-40	2716	PROM (Available only from Heath Co.)	

HEATH PART NUMBER	MAY BE REPLACED WITH	DESCRIPTION	LEAD CONFIGURATION (TOP VIEW)
444-42 444-43	SN74S47D	256 × 8-Bit PROM (Available only from Heath Co.)	
444-44 444-45	2716	PROM (Available only from Heath Co.)	
444-9001	SN74S188A	32k × 8 Read-only Memory	

APPENDIX

ASCII CHARACTERS

The characters in the shaded areas are not processed by the Computer.

7-BIT OCTAL CODE	DECIMAL CODE	HEX CODE	CHARACTERS	CONTROL KEYS	DESCRIPTION
000	0	0	NUL	@	Null, tape feed.
001	1	1	SOH	A	Start of heading.
002	2	2	STX	B	Start of text.
003	3	3	ETX	C	End of text.
004	4	4	EOT	D	End of transmission.
005	5	5	ENQ	E	Enquiry; also WRU.
006	6	6	ACK	F	Acknowledge; also RU.
007	7	7	BEL	G	Rings the bell.
010	8	8	BS	H	Backspace; also FEB, format effector backspace.
011	9	9	HT	I	Horizontal tab.
012	10	A	LF	J	Line feed: advances cursor to next line.
013	11	B	VT	K	Vertical tab (VTAB).
014	12	C	FF	L	Form feed to top of next page.
015	13	D	CR	M	Carriage return to beginning of line.
016	14	E	SO	N	Shift out.
017	15	F	SI	O	Shift in.
020	16	10	DLE	P	Data line escape.
021	17	11	DC1	Q	Device control 1: turns transmitter on (XON).
022	18	12	DC2	R	Device control 2.
023	19	13	DC3	S	Device control 3: turns transmitter off (XOFF).
024	20	14	DC4	T	Device control 4.
025	21	15	NAK	U	Negative acknowledge: also ERR (error).
026	22	16	SYN	V	Synchronous idle (SYNC).
027	23	17	ETB	W	End of transmission block.
030	24	18	CAN	X	Cancel (CANCL). Cancels current escape sequence.
031	25	19	EM	Y	End of medium.
032	26	1A	SUB	Z	Substitute.
033	27	1B	ESC	[	Escape.
034	28	1C	FS	\	File separator.
035	29	1D	GS	]	Group separator.
036	30	1E	RS	^	Record separator.
037	31	1F	US	-	Unit separator.

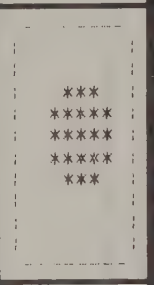
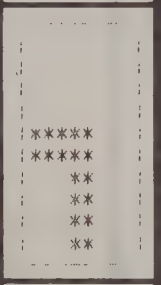
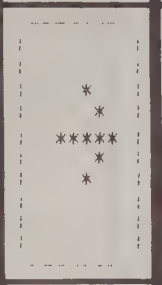
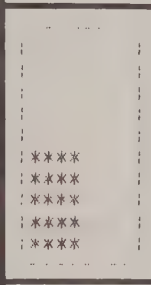
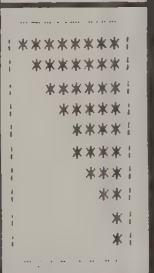
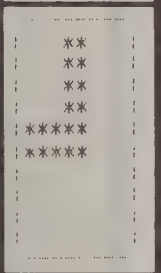
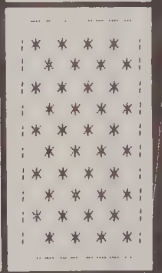
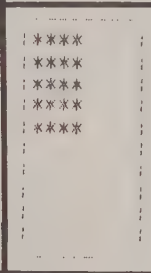
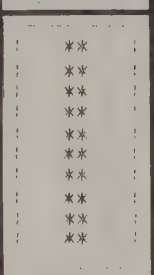
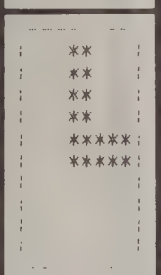
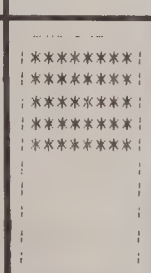
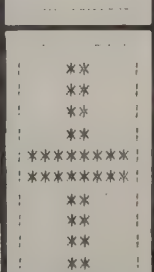
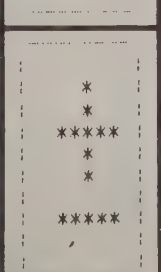
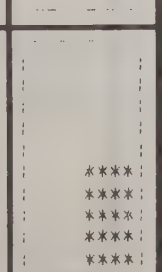
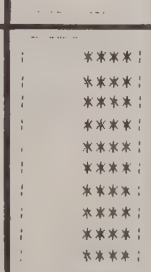
7-BIT OCTAL CODE	DECIMAL CODE	HEX CODE	CHARACTERS	DESCRIPTION
040	32	20	SP	Space.
041	33	21	!	Exclamation point.
042	34	22	"	Quotation mark.
043	35	23	#	Number sign.
044	36	24	$	Dollar sign.
045	37	25	%	Percent sign.
046	38	26	&	Ampersand.
047	39	27	'	Acute accent or apostrophe.
050	40	28	(	Open parenthesis.
051	41	29	)	Close parenthesis.
052	42	2A	*	Asterisk.
053	43	2B	+	Plus sign.
054	44	2C	,	Comma.
055	45	2D	−	Hyphen or minus sign.
056	46	2E	.	Period.
057	47	2F	/	Slash.
060	48	30	0	Number 0.
061	49	31	1	Number 1.
062	50	32	2	Number 2.
063	51	33	3	Number 3.
064	52	34	4	Number 4.
065	53	35	5	Number 5.
066	54	36	6	Number 6.
067	55	37	7	Number 7.
070	56	38	8	Number 8.
071	57	39	9	Number 9.
072	58	3A	:	Colon.
073	59	3B	;	Semicolon.
074	60	3C	<	Less than.
075	61	3D	=	Equal sign.
076	62	3E	>	Greater than.
077	63	3F	?	Question mark.

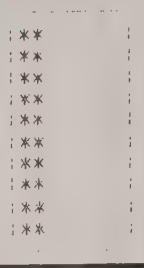
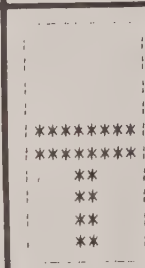
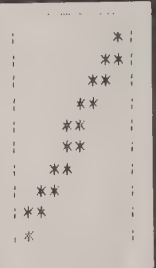
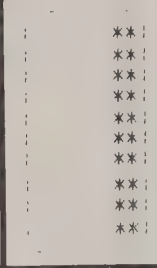
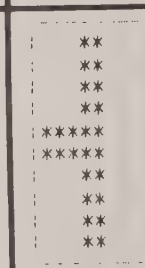
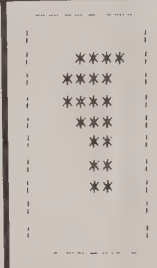
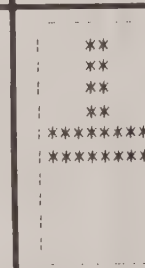
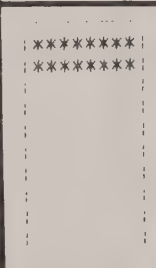
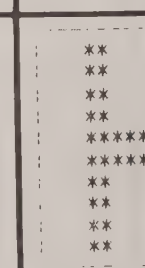
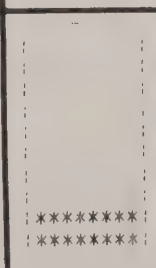
7-BIT OCTAL CODE	DECIMAL CODE	HEX CODE	CHARACTERS	DESCRIPTION	GRAPHIC SYMBOLS
100	64	40	@	At sign.	
101	65	41	A	Letter A.	
102	66	42	B	Letter B.	
103	67	43	C	Letter C.	
104	68	44	D	Letter D.	
105	69	45	E	Letter E.	
106	70	46	F	Letter F.	
107	71	47	G	Letter G.	
110	72	48	H	Letter H.	
111	73	49	I	Letter I.	
112	74	4A	J	Letter J.	
113	75	4B	K	Letter K.	
114	76	4C	L	Letter L.	
115	77	4D	M	Letter M.	
116	78	4E	N	Letter N.	
117	79	4F	O	Letter O.	
120	80	50	P	Letter P.	
121	81	51	Q	Letter Q.	
122	82	52	R	Letter R.	
123	83	53	S	Letter S.	
124	84	54	T	Letter T.	
125	85	55	U	Letter U.	
126	86	56	V	Letter V.	
127	87	57	W	Letter W.	
130	88	58	X	Letter X.	
131	89	59	Y	Letter Y.	
132	90	5A	Z	Letter Z.	
133	91	5B	[	Open brackets.	
134	92	5C	\	Reverse slash.	
135	93	5D	]	Close brackets.	
136	94	5E	↑	Up arrow/caret.	
137	95	5F	_	Underscore.	

7-BIT OCTAL CODE	DECIMAL CODE	HEX CODE	CHARACTERS	DESCRIPTION	GRAPHIC SYMBOLS
140	96	60	`	Grave accent.	
141	97	61	a	Letter a.	
142	98	62	b	Letter b.	
143	99	63	c	Letter c.	
144	100	64	d	Letter d.	
145	101	65	e	Letter e.	
146	102	66	f	Letter f.	
147	103	67	g	Letter g.	
150	104	68	h	Letter h.	
151	105	69	i	Letter i.	
152	106	6A	j	Letter j.	
153	107	6B	k	Letter k.	
154	108	6C	l	Letter l.	
155	109	6D	m	Letter m.	
156	110	6E	n	Letter n.	
157	111	6F	o	Letter o.	
160	112	70	p	Letter p.	
161	113	71	q	Letter q.	
162	114	72	r	Letter r.	
163	115	73	s	Letter s.	
164	116	74	t	Letter t.	
165	117	75	u	Letter u.	
166	118	76	v	Letter v.	
167	119	77	w	Letter w.	
170	120	78	x	Letter x.	
171	121	79	y	Letter y.	
172	122	7A	z	Letter z.	
173	123	7B	{	Left brace.	
174	124	7C	¦	Vertical bar (broken).	
175	125	7D	}	Right brace.	
176	126	7E	~	Tilde.	
177	127	7F	DEL	Delete (rubout).	

HEATHKIT®

GRAPHIC SYMBOLS

KEY (OCTAL) [DECIMAL]	SYMBOL	KEY (OCTAL) [DECIMAL]	SYMBOL	KEY (OCTAL) [DECIMAL]	SYMBOL	KEY (OCTAL) [DECIMAL]	SYMBOL
^ (136) [94]		c (143) [99]		h (150) [104]		m (155) [109]	
_ (137) [95]		d (144) [100]		i (151) [105]		n (156) [110]	
` (140) [96]		e (145) [101]		j (152) [106]		o (157) [111]	
a (141) [97]		f (146) [102]		k (153) [107]		p (160) [112]	
b (142) [98]		g (147) [103]		l (154) [108]		q (161) [113]	

KEY (OCTAL) [DECIMAL]	SYMBOL	KEY (OCTAL) [DECIMAL]	SYMBOL	KEY (OCTAL) [DECIMAL]	SYMBOL
r (162) [114]		w (167) [119]		¦ (174) [124]	
s (163) [115]		x (170) [120]		} (175) [125]	
t (164) [116]		y (171) [121]		~ (176) [126]	
u (185) [117]		z (172) [122]			
v (166) [118]		{ (173) [123]			

TRANSMITTED CODES

The following tables list the octal code or codes transmitted by the Computer when the indicated keyboard keys are pressed.

KEY	LOWER CASE	UPPER CASE
A	141	101
B	142	102
C	143	103
D	144	104
E	145	105
F	146	106
G	147	107
H	150	110
I	151	111
J	152	112
K	153	113
L	154	114
M	155	115
N	156	116
O	157	117
P	160	120
Q	161	121
R	162	122
S	163	123
T	164	124
U	165	125
V	166	126
W	167	127
X	170	130
Y	171	131
Z	172	132

ALPHABETIC KEYS

KEY	UNSHIFTED	SHIFTED	
0	060	051	)
1	061	041	!
2	062	100	@
3	063	043	#
4	064	044	$
5	065	045	%
6	066	136	↑
7	067	046	&
8	070	052	*
9	071	050	(
–	055	137	_
=	075	053	+
[	133	135	]
;	073	072	:
'	047	042	"
,	054	074	<
.	056	076	>
/	057	077	?
`	140	176	~
\	134	174	¦
{	173	175	}

NONALPHABETIC KEYS

KEYPAD KEYS	UNSHIFTED	HEATH UNSHIFTED ALTERNATE	ANSI UNSHIFTED ALTERNATE	SHIFTED
0	0	ESC ? p	ESC O p	0
1\IL	1	ESC ? q	ESC O q	ESC L (Insert Line)
2\↓	2	ESC ? r	ESC O r	ESC B (Down arrow)
3\DL	3	ESC ? s	ESC O s	ESC M (Delete Line)
4\←	4	ESC ? t	ESC O t	ESC D (Left arrow)
5\HOME	5	ESC ? u	ESC O u	ESC H (Home)
6\→	6	ESC ? v	ESC O v	ESC C (Right arrow)
7\IC	7	ESC ? w	ESC O w	ESC @ (Enter Insert Character Mode) ESC O (Exit Insert Character Mode)
8\↑	8	ESC ? x	ESC O x	ESC A (Up arrow)
9\DC	9	ESC ? y	ESC O y	ESC N (Delete Character)
•	•	ESC ? n	ESC O n	•
ENTER	RETURN	ESC ? M	ESC O M	RETURN

KEYPAD KEYS

NOTE: The shifted mode and the unshifted (or alternate) mode (if the alternate mode was selected) can be interchanged by entering ESC t or ESC u.

KEY	HEATH ESCAPE CODE	ANSI ESCAPE CODE
0	ESC ? p	ESC O p
1	ESC ? q	ESC O q
2	ESC ? r	ESC O r
3	ESC ? s	ESC O s
4	ESC ? t	ESC O t
5	ESC ? u	ESC O u
6	ESC ? v	ESC O v
7	ESC ? w	ESC O w
8	ESC ? x	ESC O x
9	ESC ? y	ESC O y
•	ESC ? n	ESC O n
ENTER	ESC ? M	ESC O M

ALTERNATE KEYPAD MODE

KEY	OCTAL CODE	ANSI CODE
RETURN	015	015
LINE FEED	012	012
BACKSPACE	010	010
SPACE BAR	040	040
TAB	011	011
DELETE	177	177
ESC	033	033

CONTROL KEYS

KEY	HEATH ESCAPE CODE	ANSI ESCAPE CODE
F1	ESC S	ESC O S
F2	ESC T	ESC O T
F3	ESC U	ESC O U
F4	ESC V	ESC O V
F5	ESC W	ESC O W
BLUE	ESC P	ESC O P
RED	ESC Q	ESC O Q
GRAY	ESC R	ESC O R

SPECIAL FUNCTION KEYS

HEATH
ESCAPE SEQUENCES

Summary Of Sequences

Escape Sequence	Mnemonic	Definition
CURSOR FUNCTIONS		
ESC H	HCUH	Cursor Home
ESC C	HCUF	Cursor Forward
ESC D	HCUB	Cursor Backward
ESC B	HCUD	Cursor Down
ESC A	HCUU	Cursor Up
ESC I	HRI	Reverse Index
ESC n	HCPR	Cursor Position Report
ESC j	HSCP	Save Cursor Position
ESC k	HRCP	Set Cursor To Previously Saved Position
ESC Y	HDCA	Direct Cursor Addressing (Same as VT52)
ERASING AND EDITING		
ESC E	HCD	Clear Display (Shift Erase)
ESC b	HBD	Erase Beginning Of Display
ESC J	HEOP	Erase To End Of Page (Erase Key)
ESC l	HEL	Erase Entire Line
ESC o	HEBL	Erase Beginning Of Line
ESC K	HEOL	Erase To End Of Line
ESC L	HIL	Insert Line
ESC M	HDL	Delete Line
ESC N	HDCH	Delete Character
ESC @	HEIM	Enter Insert Character Mode
ESC O	HERM	Exit Insert Character Mode
CONFIGURATION		
ESC z	HRAM	Reset to Power-Up Configuration
ESC r B_n	HMBR	Modify Baud Rate (B_n=; A=110, B=150, C=300, D=600, E=1200, F=1800, G=2000, H=2400, I=3600, J=4800, K=7200, L=9600)
ESC x P_s	HSM	Set Mode(s): P_s= 1 = Enable 25th line; 2 = No key click; 3 = Hold screen mode; 4 = Block cursor; 5 = Cursor off; 6 = Keypad shifted; 7 = Alternate keypad mode; 8 = Auto line feed on receipt of CR; 9 = Auto CR on receipt of line feed

ESC y P_s	HRM	Reset Mode(s): P_s= 1 = Disable 25th line 2 = Enable key click 3 = Exit hold screen mode 4 = Underscore cursor 5 = Cursor on 6 = Keypad unshifted 7 = Exit alternate keypad mode 8 = No auto line feed 9 = No auto CR
ESC <	HEAM	Enter ANSI Mode

MODES OF OPERATION

ESC [	HEHS	Enter Hold Screen Mode
ESC \	HXHS	Exit Hold Screen Mode
ESC p	HERV	Enter Reverse Video Mode
ESC q	HXRV	Exit Reverse Video Mode
ESC F	HEGM	Enter Graphics Mode
ESC G	HXGM	Exit Graphics Mode
ESC t	HEKS	Enter Keypad Shifted Mode
ESC u	HXKS	Exit Keypad Shifted Mode
ESC =	HAKM	Enter Alternate Keypad Mode
ESC >	HXAM	Exit Alternate Keypad Mode

ADDITIONAL FUNCTIONS

ESC }	HDK	Keyboard Disabled
ESC {	HEK	Keyboard Enabled
ESC v	HEWA	Wrap Around At End Of Line
ESC w	HXWA	Discard At End Of Line
ESC Z	HID	Identify As VT52 (ESC / K)
ESC]	HX25	Transmit 25th Line
ESC #	HXMP	Transmit Page

NOTE: The Terminal will transmit the following sequences, but it will not respond to them if they are received by the Terminal.

ESC S	HF1	Function Key #1 (f1)
ESC T	HF2	Function Key #2 (f2)
ESC U	HF3	Function Key #3 (f3)
ESC V	HF4	Function Key #4 (f4)
ESC W	HF5	Function Key #5 (f5)
ESC P	HF7	Function Key (BLUE)
ESC Q	HF8	Function Key (RED)
ESC R	HF9	Function Key (GRAY)

Heath Escape Sequences Defined

CURSOR FUNCTIONS

HCUH Cursor Home ESC H
Moves the cursor to the first character position on the first line (home).

HCUF Cursor Forward ESC C
Moves the cursor one character position to the right. If the cursor is at the right end of the line, it will remain there.

HCUB Cursor Backward ESC D
Moves the cursor one character position to the left (backspaces). If the cursor is at the start (left end) of a line, it will remain there.

HCUD Cursor Down ESC B
Moves the cursor down one line without changing columns. The cursor will not move past the bottom (24th) line and no scrolling will take place. Use HDCA (Direct Cursor Addressing) to move the cursor to line 25 — when line 25 is active.

HCUU Cursor Up ESC A
Moves the cursor up one line. If the cursor reaches the top line, it remains there and no scrolling occurs.

HRI Reverse Index ESC I
Moves the cursor to the same horizontal position on the preceding line. If the cursor is on the top line, a scroll down is performed.

HCPR Cursor Position Report ESC n
The Terminal reports the cursor position in the form of ESC Y line# column#.

HSCP Save Cursor Position ESC j
The present cursor position is saved so the cursor can be returned here later when given the HRCP (Set Cursor to Previously Saved Position) command.

HRCP Set Cursor To Previously Saved Position ESC k
Returns the cursor to the position where it was when it received the HSCP (Save Cursor Position) command.

HDCA Direct Cursor Addressing ESC Y
Moves the cursor to a position on the scceen by entering the escape code, the ASCII character which represents the line number, and the ASCII character which represents the column number.

The first line and the left column are both 32_{10} (the smallest value of the printing characters) and increase from there. Since the lines are numbered from 1 to 25 (from top to bottom) and the columns from 1 to 80 (from left to right), you must add the proper line and column numbers to 31_{10}. Then convert these decimal numbers to their equivalent ASCII characters and enter them in the following order:

ESC Y line # (ASCII character) column # (ASCII character)

If the line number entered is too high, the cursor will not move. If the column number is too high, the cursor will move to the end of the line.

This is the only way to move the cursor to the 25th line, but the 25th line must first be enabled.

ERASING AND EDITING

HCD Clear Display (Shift Erase) ESC E
Erases the entire screen, fills the screen with spaces, and places the cursor in the home position.

HBD Erase Beginning of Display ESC b
Erases from the start of the screen to the cursor, and includes the cursor position.

HEOP Erase to End Of Page (Erase Key) ESC J
Erases all the information from the cursor (including the cursor position) to the end of the page.

HEL ERASE Entire Line ESC l
Erases all of the line, including the cursor position.

HEBL Erase Beginning of Line ESC o
Erases from the beginning of the line to the cursor, and includes the cursor position.

HEOL Erase to End Of Line ESC K
Erases from the cursor (including the cursor position) to the end of the line.

HIL Insert Line ESC L
Inserts a new blank line by moving the line that the cursor is on, and all following lines, down one line. Then the cursor is moved to the beginning of the new blank line.

HDL Delete Line ESC M
Deletes the contents of the line that the cursor is on, places the cursor at the beginning of the line, moves all the following lines up one line, and adds a blank line at line 24.

HDCH Delete Character ESC N
Deletes the character at the cursor position and shifts any existing text that is to the right of the cursor one character position to the left.

HEIM Enter Insert Character Mode ESC @
Lets you insert characters or words into text already displayed on the screen. As you type in new characters, existing text to the right of the cursor shifts to the right. As each new character is inserted, the character at the end of the line is lost.

HERM Exit Insert Character Mode ESC O
Exits from the insert character mode.

CONFIGURATION

HRAM Reset to Power-Up Configuration ESC z
Nullifies all previously set escape modes and returns to the power-up configuration.

HMBR Modify Baud Rate ESC r Bn
Modifies the baud rate, where Bn equals:
A=110, B=150, C=300, D=600,
E=1200, F=1800, G=2000, H=2400,
I=3600, J=4800, K=7200, L=9600

HSM Set Mode(s) ESC x P_s
Sets the following modes, where P_s equals:
1=enable 25th line
2=no key click
3=hold screen mode
4=block cursor
5=cursor off
6=keypad shifted
7=alternate keypad mode
8=auto line feed on receipt of CR
9=auto CR on receipt of line feed

HRM Reset Mode(s) ESC y P_s
Resets special modes, where P_s equals:
1=disable 25th line
2=enable key click
3=exit hold screen mode
4=underscore cursor
5=cursor on
6=keypad unshifted
7=exit alternate keypad mode
8=no auto line feed
9=no auto CR

HEAM Enter ANSI Mode ESC <
Enters the ANSI mode.

MODES OF OPERATION

HEHS Enter Hold Screen Mode ESC [
Controls when new information is printed on the screen.
— Type the SCROLL key and a new line of information will be printed on the bottom line. The top line will scroll off.
— Type SHIFT SCROLL and a whole new page of text will scroll onto the screen and stop as the old page scrolls up and off the screen.

HXHS Exit Hold Screen Mode ESC
Exits the hold screen mode.

HERV Enter Reverse Video Mode ESC p
Enters the reverse video mode so that characters are displayed as black characters on a white background.

HXRV Exit Reverse Video Mode ESC q
Exits the reverse video mode.

HEGM Enter Graphics Mode ESC F
Enters the graphics mode to display any of the 33 special symbols (26 lower-case keys and seven other keys) that correspond to the graphic symbols.

HXGM Exit Graphics Mode ESC G
Exits the graphics mode and returns to the display of normal characters.

HEKS Enter Keypad Shifted Mode ESC t
Inverts the normal and shifted functions of the keypad. Now, if you hold down the SHIFT key, you will get a normally unshifted character.

HXKS Exit Keypad Shifted Mode ESC u
Exits the keypad shifted mode.

HAKM Enter Alternate Keypad Mode ESC =
Enters the alternate keypad mode, which will then allow the keyboard keys to transmit the following escape codes instead of the normal ones.

KEY	ESCAPE CODE
0	ESC ? p
1	ESC ? q
2	ESC ? r
3	ESC ? s
4	ESC ? t
5	ESC ? u
6	ESC ? v
7	ESC ? w
8	ESC ? x
9	ESC ? y
.	ESC ? n
ENTER	ESC ? M

These special escape codes are user defined and must be recognized by your software.

HXAM Exit Alternate Keypad Mode ESC >
Exits the alternate keypad mode and returns to the transmission of normal character codes.

ADDITIONAL FUNCTIONS

HDK Keyboard Disabled ESC }
Inhibits the output of the keyboard.

HEK Keyboard Enabled ESC {
Enables the keyboard after it was inhibited by an HDK (Keyboard Disabled) command.

HEWA Wrap Around at End of Line ESC v
The 81st character on a line is automatically placed in the first character position on the next line. The page scrolls up if necessary.

HXWA Discard at End of Line ESC w
After the 80th character in a line, the characters overprint. Therefore, only the last character received will be displayed in position 80.

HID Identify as VT52 (ESC / K) ESC Z
The Terminal responds to the interrogation with ESC / K to indicate that it can perform as VT52.

HX25 Transmit 25th Line ESC]
Transmits the 25th line. (The computer requires a special routine to use this feature.)

HXMP Transmit Page ESC #
Transmits lines 1 through 24. (The computer requires a special routine to use this feature.)

HF1 Function Key #1 (F1) ESC S
Transmits a unique escape code to perform a user-defined function. The Terminal will not respond to this code if it is received.

HF2 Function Key #2 (F2) ESC T
Same as above.

HF3 Function Key #3 (F3) ESC U
Same as above.

HF4 Function Key #4 (F4) ESC V
Same as above.

HF5 Function Key #5 (F5) ESC W
Same as above.

HF7 Function Key Blue ESC P
Same as above.

HF8 Function Key Red ESC Q
Same as above.

HF9 Function Key Gray ESC R
Same as above.

ANSI ESCAPE SEQUENCES

Summary Of Sequences

NOTES:

1. In the ANSI mode, the Terminal recognizes and responds only to escape sequences whose syntax and semantics are in accordance with ANSI specifications.

2. "Default" is a value that is assumed when no explicit value, or a value of zero, is specified.

3. P_n — Numeric Parameter. Any decimal value may be substituted for P_n.

4. P_s — Selective Parameter. Any decimal number that is taken from a list and used to select a subfunction. You can select several subfunctions at once by putting one number after another but separating them with delimiters (semicolons).

   Example: To turn off the key click (ESC [> 2 h) and turn on the block cursor (ESC [> 4 h), type:

   ESC [> 2;4 h

Escape Sequence	Mnemonic	Definition
CURSOR FUNCTIONS		
ESC [H or ESC [Ø;Ø H or ESC [1;1 H or ESC [f or ESC [Ø;Ø f or ESC [1;1 f	CUP or HVP	Cursor Home
ESC [P_n C	CUF	Cursor Forward
ESC [P_n D	CUB	Cursor Backward
ESC [P_n B	CUD	Cursor Down
ESC [P_n A	CUU	Cursor Up
ESC M	RI	Reverse Index
ESC [n	CPR	Cursor Position Report
ESC [s	PSCP	Save Cursor Position
ESC [u	PRCP	Set Cursor Position
ESC [P_l;P_c H or ESC [P_l;P_c f	CUP	Direct Cursor Addressing

ERASING AND EDITING

ESC [2 j	ED	Clear Display (Shift Erase)
ESC [1 j	ED	Erase Beginning Of Display
ESC [j or ESC [ϕ J	ED	Erase To End Of Page (Erase Key)
ESC [2 k	EL	Erase Entire Line
ESC [1 K	EL	Ease Beginning Of Line
ESC [K or ESC [ϕ K	EL	Erase To End Of Line
ESC [P_n L	IL	Insert Line
ESC [P_n M	DL	Delete Line
ESC [P_n P	DCH	Delete Character
ESC [4 h	IRM	Insert/Replacement (Insert character) Mode On
ESC [4 l	IRM	Insert/Replacement (Insert Character) Mode Off

CONFIGURATION

ESC [z	PRAM	Reset To Power-Up Configuration
ESC [P_n r	PMBR	Modify Baud Rate (P_n=; 1=110, 2=150, 3=300, 4=600, 5=1200, 6=1800, 7=2000, 8=2400, 9=3600, 10=4800, 11=7200, 12=9600)
ESC [> P_n h	SM	Set Mode(s): P_n= 1 = Enable 25th line 2 = No key click 3 = Hold screen mode 4 = Block cursor 5 = Cursor off 6 = Keypad shifted 7 = Alternate Keypad mode 8 = Auto line feed on receipt of CR 9 = Auto CR on receipt of line feed
ESC [> P_n l	RM	Reset Mode(s): P_n= 1 = Disable 25th line 2 = Enable key click 3 = Exit hold screen mode 4 = Underscore cursor 5 = Cursor on 6 = Keypad unshifted 7 = Exit alternate keypad mode 8 = No auto line feed 9 = No auto CR
ESC [? 2 h	PEHM	Enter Heath Mode

MODES OF OPERATION

ESC [7 m	SGR	Enter Reverse Video Mode
ESC [m or ESC [ϕ m	SGR	Exit Reverse Video Mode
ESC [> 7 h	SM	Enter Alternate Keypad Mode (ESC =)*
ESC [> 7 l	RM	Exit Alternate Keypad Mode (ESC >)*
ESC [10 m	SGR	Enter Graphics Mode
ESC [11 m	SGR	Exit Graphics Mode

*These escape codes may be used, but are not recommended.

ADDITIONAL FUNCTIONS

ESC [2 h	SM	Keyboard Disabled
ESC [2 l	RM	Keyboard Enabled
ESC [? 7 h	SM	Wrap Around At End Of Line
ESC [? 7 l	RM	Discard At End Of Line
ESC [q	PX25	Transmit 25th Line
ESC [p	PXMT	Transmit Page

NOTE: The Terminal will transmit the following functions, but it will not respond to them if they are received by the Terminal.

ESC O S	SS3	Function Key #1 (F1)
ESC O T	SS3	Function Key #2 (F2)
ESC O U	SS3	Function Key #3 (F3)
ESC O V	SS3	Function Key #4 (F4)
ESC O W	SS3	Function Key #5 (F5)
ESC O P	SS3	Function Key (BLUE)
ESC O Q	SS3	Function Key (RED)
ESC O R	SS3	Function Key (GRAY)

ANSI Mode Summary

The ANSI controls SET MODE (SM) and RESET MODE (RM) are shown on the previous page. The following table shows all parameters which may be set or reset using the SM and RM control sequences.

The control sequence for SET MODE is: ESC [P_s h.
The control sequence for RESET MODE is: ESC [P_s l.

	MODE	P_s	SET (SM)	RESET (RM)
ANSI	KAM	2	Keyboard Disabled	Keyboard Enabled
	IRM	4	Insert Character Mode On	Insert Character Mode OFF
	LNM	2Ø	New Line Mode (Auto Line Feed On CR)	New Line Mode Off
HEATH	L25	>1	Display 25th Line	Disable 25th Line
	KCL	>2	Disable Key Click	Enable Key Click
	HSM	>3	Enable Hold Screen Mode	Disable Hold Screen Mode
	CBL	>4	Blinking Block Cursor	Blinking Underscore Cursor
	CDE	>5	Cursor Off	Cursor On
	KSH	>6	Keypad Shifted	Keypad Unshifted
	KAM	>7	Keypad Alternate Mode	Keypad Normal Mode
	ALF	>8	Auto Line Feed On Return	No Auto Line Feed
	ACR	>9	Auto CR On Line Feed	No Auto CR On Line Feed
	HMD	?2	Enter Heath Mode	N/A
	WAR	?7	Wrap Around At End Of Line	Discard Past End Of Line

ANSI modes which are always considered to be in either the SET or the RESET state, and those which do not apply to this product are as follows:

CRM	Control Representation Mode	RESET
EBM	Editing Boundary Mode	RESET
ERM	Erasure Mode	SET
FEAM	Format Effector Action Mode	RESET
FETM	Format Effector Transfer Mode	RESET
GATM	Guarded Area Transfer Mode	RESET
HEM	Horizontal Editing Mode	RESET
MATM	Multiple Area Transfer Mode	N/A
PUM	Positioning Unit Mode	RESET
SATM	Selected Area Transfer Mode	SET
SRTM	Status Reporting Transfer Mode	N/A
TSM	Tabulation Stop Mode	N/A
TTM	Transfer Termination Mode	SET
VEM	Vertical Editing Mode	RESET
SEM	Set Editing Extent Mode	Edit In Line

ANSI Escape Sequences Defined

NOTES:

1. In the ANSI mode, the Terminal recognizes and responds only to escape sequences whose syntax and semantics are in accordance with ANSI specifications.

2. "Default" is a value that is assumed when no explicit value, or a value of zero, is specified.

3. P_n — Numeric Parameter. Any decimal number that is substituted for P_n.

4. P_s — Selective Parameter. Any decimal number that is taken from a list and used to select a subfunction. You can select several subfunctions at once by putting one number after another but separating them with delimiters (semicolons).

CURSOR FUNCTIONS

CUP Cursor Position — **ESC [H** or **ESC [ϕ;ϕ H** or
or — **ESC [1;1 H**
HVP Horizontal & Vertical Position — **ESC [f** or **ESC [ϕ;ϕ f)** or
ESC [1;1 f

Moves the cursor to the position specified by the parameters. The first parameter specifies the line number and the second parameter specifies the column number. A parameter of zero is considered to be one. If no parameter is given, the cursor is placed in the home position.

Default Value: 1

CUF Cursor Forward ESC [P_n C
Moves the cursor to the right the number of characters determined by the value of P_n. If this number is zero or one, the cursor moves one position. The cursor stops at the right margin.

Default Value: 1

CUB Cursor Backward ESC [P_n D
Moves the cursor to the left the number of characters determined by the value of P_n. If this number is zero or one, the cursor moves one position. The cursor stops at the left margin.

Default Value: 1

CUD Cursor Down ESC [P_n B
Moves the cursor downward without changing columns. The number of lines moved is determined by the value of P_n. If this number is zero or one, the cursor moves down one line. The cursor will stop at line 24. Direct Cursor Addressing must be used to move to line 25.

CUU Cursor Up ESC [P_n A
Moves the cursor upward without changing columns. The number of lines moved is determined by the value of P_n. If this number is zero or one, the cursor moves up one line. The cursor will stop at the top line.

Default Value:1

RI Reverse Index ESC M
Moves the cursor to the same position on the preceeding line.

CPR Cursor Position Report ESC [n
The Terminal reports the cursor position in the form of ESC [P_l;P_c R.

PSCP Save Cursor Position ESC [s
The present cursor position is remembered so the cursor can be returned here later when given the PRCP (Return to Previously Saved Position) command.

PRCP Set Cursor to Previously Saved Position ESC [u
Returns the cursor to the position where it was when it received the PSCP (Save Cursor Position) command.

CUP Direct Cursor Addressing ESC [P_l;P_c H or
ESC [P_l;P_c f
Same as CUP and HVP above. If the line number (P_l) entered is too high, the cursor will not move. If the column number (P_c) is too high, the cursor will move to the end of the line.
This is the only way to move the cursor to the 25th line, but the 25th line must first be enabled.
To move the cursor home, enter ϕ;ϕ or l;l or do not enter any values.
Default Values: 1

ERASING AND EDITING

ED Erase In Display ESC [P_s J
Erases some or all of the characters in the display according to the value of P_s.

P_s	Means
ϕ	Erases from the cursor to the end of the screen and includes the cursor position.
1	Erases from the start of the screen to the cursor and includes the cursor position.
2	Erases all of the screen and the cursor goes to the HOME position.

Default Value: 0

EL Erase In Line ESC [P_s K
Erases some or all of the characters in the cursor line according to the value of P_s.

P_s	Means
ϕ	Erases from the cursor to the end of the line and includes the cursor position.
1	Erases from the start of the line to the cursor and includes the cursor position.
2	Erases all of the line including the cursor position.

Default Value: 0

IL Insert Line ESC [P_n L
Inserts one or more blank lines (depending on the value of P_n) by moving the line that the cursor is on and all the following lines down P_n lines. Then the cursor is moved to the beginning of the new blank line.

DL Delete Line ESC [P_n M
Deletes the line of characters that the cursor is in, and other following lines if P_n is greater than one. The remaining lines below the deleted area then move up the number of lines that were deleted. The cursor is placed at the beginning of the next line.
Default Value: 1

DCH Delete Character ESC [P_n P
Deletes the characters at the cursor position, and other positions on the cursor line to the right of the cursor if P_n is greater than one. Any remaining characters to the right of the deleted characters then move left the number of characters that were deleted.
Default Value: 1

IRM Insert/Replacement Mode ON ESC [4 h
Lets you insert characters or words into text already displayed on the screen. As new characters are entered, existing text to the right of the cursor shifts to the right. As each character is inserted, the character at the end of the line is lost.

IRM Insert/Replacement Mode OFF ESC [4 l
Exits from the IRM ON mode.

CONFIGURATION

PRAM Reset to Power-Up Configuration ESC [z
Nullifies all previously set escape modes and returns to the power-up configuration.

PMBR Modify Baud Rate ESC [P_n r
Modifies the baud rate, where P_n equals:
1=110, 2=150, 3=300, 4=600, 5=1200,
6=1800, 7=2000, 8=2400, 9=3600, 10=4800,
11=7200, 12=9600

SM Set Mode(s), ESC [> P_s h
Sets the following modes, where P_n equals:
1=enable 25th line
2=no key click
3=hold screen mode
4=block cursor
5=cursor off
6=keypad shifted
7=alternate keypad mode
8=auto line feed on receipt of CR
9=auto CR on receipt of line feed

Can set one or more modes as determined by the parameter string P_s;P_s;P_s, etc.
Default Value: None

RM Reset Mode(s) ESC [> P_s l
Resets special modes, where P_s equals:

1=disable 25th line
2=enable key click
3=exit hold screen mode
4=underscore cursor
5=cursor on
6=keypad unshifted
7=exit alternate keypad mode
8=no auto line feed
9=no auto CR

Can reset one or more modes as determined by the parameter string P_s;P_s;P_s, etc.
Default Value: None

PEHM Enter Heath Mode ESC [? 2 l
Enters the Heath mode.

MODES OF OPERATION

SM Enter Hold Screen Mode ESC [> 3 h
Controls when new information is printed onto the screen.

— Type the SCROLL key and a new line of information will be printed on the bottom line. The top line will scroll off.
— Type SHIFT SCROLL and a whole new page of text will scroll onto the screen and stop as the old page scrolls up and off the screen.

RM Exit Hold Screen Mode ESC [> 3 l
Exits the hold screen mode.

SGR Enter Reverse Video Mode ESC [7 m
Enters the reverse video mode so that characters are displayed as black characters on a white background.

SGR Exit Reverse Video Mode ESC [m or ESC [0 m
Exits the reverse video mode.

SM Enter Keypad Shifted Mode ESC [> 6 h
Inverts the normal and shifted functions of the keypad. Now if you hold down the SHIFT key, you will get a normally unshifted character.

RM Exit Keypad Shifted Mode ESC [> 6 l
Exits the keypad shifted mode.

SM Enter Alternate Keypad Mode ESC = or ESC [> 7 h
Allows you to enter the alternate keypad mode, which will then transmit the following escape codes instead of the normal ones.

KEY	ESCAPE CODE
∅	ESC O p
1	ESC O q
2	ESC O r
3	ESC O s
4	ESC O t
5	ESC O u
6	ESC O v
7	ESC O w
8	ESC O x
9	ESC O y
•	ESC O n
ENTER	ESC O M

These special escape codes are user defined and must be recognized by your software.

RM Exit Alternate Keypad Mode ESC > or ESC [> 7 l
Exits the alternate keypad mode and returns to the transmission of normal character codes.

ADDITIONAL FUNCTIONS

SM Keyboard Disabled ESC [2 h
Inhibits the output of the keyboard. To activate the keyboard, send the "enable keyboard" escape sequence from the computer or reset the Terminal.

RM Keyboard Enabled ESC [2 l
Enables the keyboard after it was inhibited by an SM (Keyboard Disabled) command.

SM Wrap Around At End Of Line ESC [? 7 h
81st character on a line is automatically placed in the first character position on the next line. The page scrolls up if necessary and permitted.

RM Discard At End Of Line ESC [? 7 l
After the 80th character in a line, the characters overprint. Therefore, only the last character received will be displayed in position 80.

PX25 Transmit 25th Line ESC [q
Transmits the 25th line.

PXMT Transmit Page ESC [p
Transmits lines 1 through 24. (The computer requires a special routine to use this feature.)

SS3 Function Key #1 (F1) ESC O S
Transmits a unique escape code to perform a user-defined function. The Terminal will not respond to this code if it is received.

SS3 Function Key #2 (F2) ESC O T
Same as above.

SS3 Function Key #3 (F3) ESC O U
Same as above.

SS3 Function Key #4 (F4) ESC O V
Same as above.

SS3 Function Key #5 (F5) ESC O W
Same as above.

SS3 Function Key (Blue) ESC O P
Same as above.

SS3 Function Key (Red) ESC O Q
Same as above.

SS3 Function Key (Gray) ESC O R
Same as above.

THE FUNCTIONS OF A COMPUTER

This section of the Manual introduces certain basic computer concepts. It provides background information and definitions which will be useful.

A TYPICAL COMPUTER SYSTEM

A typical digital computer consists of:

a) A central processor unit (CPU)
b) A memory
c) Input/output (I/O) ports

The memory serves as a place to store **instructions,** the coded information that directs the activities of the CPU, and **data**, the coded information processed by the CPU. A group of logically related instructions stored in memory is referred to as a **program**. The CPU "reads" each instruction from memory in a logically determined sequence, and uses it to initiate processing actions. If the program sequence is coherent and logical, processing the program produces intelligible and useful results.

The memory is also used to store the data to be manipulated, as well as the instructions that direct manipulation. The program must be organized such that the CPU does not read a non-instruction word when it expects to see an instruction. The CPU can rapidly access data stored in memory, but often the memory is not large enough to store the data required for a particular application. This problem can be resolved by providing the computer with one or more **input ports.** The CPU can address these ports and input the data contained there. The addition of input ports enables the computer to receive information from external equipment (such as a magnetic tape console or floppy disk) at high rates of speed and in large volumes.

A computer also requires one or more **output ports** that permit the CPU to communicate the result of its processing to the outside world. The output may go to a display, for use by a human operator, to a peripheral device that produces "hard copy," such as a line printer, to a peripheral storage device, such as a floppy disk unit, or the output may constitute process control signals that direct the operations of another system, such as an automated assembly line. Like input ports, output ports are addressable. The input and output ports together permit the processor to communicate with the outside world.

The CPU unifies the system. It controls the functions performed by the other components. The CPU fetches instructions from memory, decodes their binary contents and executes them. It also references memory and I/O ports as necessary in the execution of instructions. In addition, the CPU recognizes and responds to certain external control signals, such as interrupt and wait requests. The functional units within a CPU that enable it to perform these functions are described below.

THE ARCHITECTURE OF A CPU

A typical central processor unit (CPU) consists of the following interconnected functional units:

- Registers
- Arithmetic/Logic Unit (ALU)
- Control Circuitry

Registers are temporary storage units within the CPU. Some registers, such as the program counter and instruction register, have dedicated uses. Other registers, such as the accumulator, are for general-purpose use.

Accumulator

The accumulator usually stores one of the operands to be manipulated by the ALU. A typical instruction might direct the ALU to add the contents of some other register to the contents of the accumulator and store the result in the accumulator itself. In general, the accumulator is both a source (operand) and a destination (result) register.

Often a CPU includes a number of additional general purpose registers used to store operands or intermediate data. The availability of general-purpose registers eliminates the need to "shuffle" intermediate results back and forth between memory and the accumulator, thus improving processing speed and efficiency.

Portions of this section of the Manual are reprinted by permission of Intel Corporation (Copyright 1976).

Program Counter (Jumps, Subroutines and the Stack)

The instructions that make up a program are stored in the system's memory. The central processor references the contents of memory in order to determine what action is appropriate. This means the processor must know which location contains the next instruction.

Each of the locations in memory is numbered to distinguish it from all other locations in memory. The number that identifies a memory location is called its **address.**

The processor maintains a counter that contains the address of the next program instruction. This register is called the **program counter**. The processor updates the program counter by adding "1" to the counter each time it fetches an instruction. Therefore, the program counter is always current (pointing to the next instruction).

The programmer therefore stores his instructions in numerically adjacent addresses, so the lower addresses contain the first instructions to be executed and the higher addresses contain later instructions. The only time the programmer may violate this sequential rule is when an instruction in one section of memory is a **jump** instruction to another section of memory.

A jump instruction contains the address of the instruction which is to follow it. The next instruction may be stored in any memory location, as long as the programmed jump specifies the correct address. During the execution of a jump instruction, the processor replaces the contents of its program counter with the address embodied in the instruction. Thus, the logical continuity of the program is maintained.

A special kind of program jump occurs when the stored program "**calls**" a subroutine. In this kind of jump, the processor is required to "remember" the contents of the program counter at the time the call occurs. This enables the processor to resume execution of the main program when it is finished with the last instruction of the subroutine.

A **subroutine** is a program within a program. Usually it is a general-purpose set of instructions that must be executed repeatedly in the course of a main program. Routines which calculate the square, the sine, or the logarithm of a program variable are good examples of functions often written as subroutines. Other examples are programs designed for inputting or outputting data to a particular peripheral device.

The processor has a special way of handling subroutines, in order to insure an orderly return to the main program. When the processor receives a call instruction, it increments the program counter and stores the counter's contents in a reserved memory area known as the **stack**. The stack thus saves the address of the instruction to be executed after the subroutine is completed. Then the processor loads the address specified in the call into its program counter. The next instruction fetched is therefore the first step of the subroutine.

The last instruction in any subroutine is a **return**. Such an instruction need specify no address. When the processor fetches a return instruction, it simply replaces the current contents of the program counter with the address on the top of the stack. This causes the processor to resume execution of the program at the point immediately following the original call instruction.

Subroutines are often **nested**; that is, one subroutine will sometimes call a second subroutine. The second may call a third, and so on. This is perfectly acceptable, as long as the processor has enough capacity to store the necessary return addresses, and the logical provision for doing so. In other words, the maximum depth of nesting is determined by the depth of the stack itself. If the stack has space for storing three return addresses, then three levels of subroutine nesting may be accommodated.

Processors have different ways of maintaining stacks. Some have facilities for the storage of return addresses built into the processor itself. Other processors use a reserved area of external memory as the stack and simply maintain a **pointer** register which contains the address of the most recent stack entry. The external stack allows virtually unlimited subroutine nesting. In addition, if the processor provides instructions that cause the contents of the accumulator and other general-purpose registers to be "pushed" onto the stack or "popped" off the stack via the address stored in the stack pointer, multi-level interrupt processing (described later in this section) is possible. The status of the processor (for example, the contents of all the registers) can be saved in the stack when an interrupt is accepted and then restored after the interrupt has been serviced. This ability to save the processor's status at any given time is possible even if an interrupt service routine, itself, is interrupted.

Instruction Register and Decoder

Every computer has a **word length** characteristic of that machine. A computer's word length is usually determined by the size of its internal storage elements and interconnecting paths (referred to as **buses**); for example, a computer whose registers and buses can store and transfer eight bits of information has a characteristic word length of eight bits and is referred to as an 8-bit parallel processor. An 8-bit parallel processor generally finds it most efficient to deal with 8-bit binary fields, and the memory associated with such a processor is therefore organized to store eight bits in each addressable memory location. Data and instructions are stored in memory as 8-bit binary numbers, or as numbers that are integral multiples of eight bits: 16 bits, 24 bits, and so on. This characteristic 8-bit field is often referred to as a **byte**.

Each operation the processor can perform is identified by a unique byte of data known as an **instruction code** or **operation code**. An 8-bit word used as an instruction code can distinguish between 256 alternative actions, more than adequate for most processors.

The processor fetches an instruction in two distinct operations. First, the processor transmits the address in its program counter to the memory. Then the memory returns the addressed byte to the processor. The CPU stores this instruction byte in the **instruction register**, and uses it to direct activities during the remainder of the instruction execution.

The mechanism by which the processor translates an instruction code into specific processing actions requires a more elaborate explanation than is given here. The concept, however, should be intuitively clear to any logic designer. The eight bits stored in the instruction register can be decoded and used to selectively activate one of a number of output lines, in this case up to 256 lines. Each line represents a set of activities associated with execution of a particular instruction code. The enabled line can be combined with selected timing pulses to develop electrical signals that can then be used to initiate specific actions. This translation of code into action is performed by the **instruction decoder** and the associated control circuitry.

An 8-bit instruction code is often sufficient to specify a particular processing action. There are times, however, when execution of the instruction requires more information than eight bits can convey.

One example of this is when the instruction references a memory location. The basic instruction code identifies the operation to be performed, but cannot specify the object address as well. In a case like this, a two- or three-byte instruction must be used. Successive instruction bytes are stored in sequentially adjacent memory locations, and the processor performs two or three fetches in succession to obtain the full instruction. The first byte retrieved from memory is placed in the processor's instruction register, and subsequent bytes are placed in temporary storage; the processor then proceeds with the execution phase. Such an instruction is referred to as **variable length**.

Address Register(s)

A CPU may use a register or register pair to hold the address of a memory location to be accessed for data. If the address register is **programmable**, (for example, if there are instructions that allow the programmer to alter the contents of the register) the program can "build" an address in the address register prior to executing a **memory reference** instruction (for example, an instruction that reads data from memory, writes data to memory, or operates on data stored in memory).

Arithmetic/Logic Unit (ALU)

All processors contain an arithmetic/logic unit, often referred to simply as the **ALU**. The ALU, as its name implies, is that portion of the CPU hardware which performs the arithmetic and logical operations on the binary data.

The ALU must contain an **adder** capable of combining the contents of two registers in accordance with the logic of binary arithmetic. This provision permits the processor to perform arithmetic manipulations on the data it obtains from memory and from its other inputs.

Using only the basic adder, a capable programmer can write routines which will subtract, multiply and divide, giving the machine complete arithmetic capabilities. In practice, however, most ALU's provide other built-in functions, including hardware subtraction, Boolean logic operations, and shift capabilities.

The ALU contains **flag bits** which specify certain conditions that arise in arithmetic and logical manipulations. Flags typically include **carry, zero, sign**, and **parity**. It is possible to program jumps which are conditionally dependent on the status of one or more flags. Thus, for example, the program may be designed to jump to a special routine if the carry bit is set following an addition instruction.

Control Circuitry

The control circuitry is the primary functional unit within a CPU. Using clock inputs, the control circuitry maintains the proper sequence of events required for any processing task. After an instruction is fetched and decoded, the control circuitry issues the appropriate signals (to units both internal and external to the CPU) for initiating the proper processing action. Often the control circuitry is capable of responding to external signals, such as an interrupt or wait request. An **interrupt** request causes the control circuitry to temporarily interrupt main program execution, jump to a special routine to service the interrupting device, then automatically return to the main program. A **wait** request is often issued by a memory or I/O element that operates slower than the CPU. The control circuitry will idle the CPU until the memory or I/O port is ready with the data.

COMPUTER OPERATIONS

There are certain operations basic to almost any computer. A sound understanding of these basic operations is a necessary prerequisite to examining the specific operations of a particular computer.

Timing

The activities of the central processor are cyclical. The processor fetches an instruction, performs the operations required, fetches the next instruction, and so on. This orderly sequence of events requires precise timing, and the CPU therefore requires a free-running oscillator clock that furnishes the reference for all processor actions. The combined fetch and execution of a single instruction is referred to as an **instruction cycle**. The portion of a cycle identified with a clearly defined activity is called a **state**. And the interval between pulses of the timing oscillator is referred to as a **clock period**. As a general rule, one or more clock periods are necessary for the completion of a state, and there are several states in a cycle.

Instruction Fetch

The first state(s) of any instruction cycle is dedicated to fetching the next instruction. The CPU issues a read signal and the contents of the program counter are sent to memory, which responds by returning the next instruction word. The first byte of the instruction is placed in the instruction register. If the instruction consists of more than one byte, additional states are required to fetch each byte of the instruction. When the entire instruction is present in the CPU, the program counter is incremented (in preparation for the next instruction fetch) and the instruction is decoded. The operation specified in the instruction will be executed in the remaining states of the instruction cycle. The instruction may call for a memory read or write, an input or output and/or internal CPU operation, such as a register-to-register transfer or an add-registers operation.

Memory Read

An instruction **fetch** is merely a special memory read operation that brings the instruction to the CPU's instruction register. The instruction fetched may then call for data to be read from memory into the CPU. The CPU again issues a read signal and sends the proper memory address; memory responds by returning the requested word. The data received is placed in the accumulator or one of the other general-purpose registers (not the instruction register).

Memory Write

A memory write operation is similar to a read except for the direction of data flow. The CPU issues a write signal, sends the proper memory address, then sends the data word to be written into the addressed memory location.

Wait

As previously stated, the activities of the processor are timed by a master clock oscillator. The clock period determines the timing of all processing activity.

The speed of the processing cycle is limited by the memory's **access time**. Once the processor has sent a read address to memory, it cannot proceed until the memory has had time to respond. Most memories are capable of responding much faster than the processing cycle requires. A few, however, cannot supply the addressed byte within the minimum time established by the processor's clock.

Therefore, a processor contains a synchronization provision, which permits the memory to request a **wait state**. When the memory receives a read or write enable signal, it places a request signal on the processor's READY line, causing the CPU to idle temporarily. After the memory has had time to respond, it frees the processor's READY line, and the instruction cycle proceeds.

Input/Output

Input and Output operations are similar to memory read and write operations with the exception that a peripheral I/O device is addressed instead of a memory location. The CPU issues the appropriate input or output control signal, sends the proper device address, and either receives the data being input or sends the data to be output.

Data can be input/output in either parallel or serial form. All data within a digital computer is represented in binary coded form. A binary data word consists of a group of bits; each bit is either a one or a zero. **Serial** I/O consists of transferring one bit at a time on a single line. Naturally, serial I/O is much slower, but it requires considerably less hardware than does parallel I/O.

Interrupts

Interrupt provisions are included on many central processors as a means of improving the processor's efficiency. Consider the case of a computer processing a large volume of data, portions of which are to be output to a printer. The CPU can output a byte of data within a single machine cycle but it may take the printer the equivalent of many machine cycles to actually print the character specified by the data byte. The CPU could then remain idle, waiting until the printer can accept the next data byte. If an interrupt capability is implemented on the computer, the CPU can output a data byte, then return to data processing. When the printer is ready to accept the next data byte, it can request an interrupt. When the CPU acknowledges the interrupt, it suspends main program execution and automatically branches to a routine that will output the next data byte. After the byte is output, the CPU continues with main program execution. Note that this is, in principle, quite similar to a subroutine call, except the jump is initiated externally rather than by the program.

More complex interrupt structures are possible in which several interrupting devices share the same processor but have different priority levels. Interruptive processing is an important feature that enables maximum utilization of a processor's capacity for high system throughput.

INSTRUCTION SET

A computer, no matter how sophisticated, can only do what it is "told" to do. A computer is told what to do via a series of coded instructions referred to as a **program**. The realm of the programmer is referred to as **software**, in contrast to the **hardware** that comprises the actual computer equipment. A computer's software refers to all of the programs that have been written for that computer.

When a computer is designed, the engineers provide the Central Processing Unit (CPU) with the ability to perform a particular set of operations. The CPU is designed such that a specific operation is performed when the CPU control logic decodes a particular instruction. Consequently, the operations that can be performed by a CPU define the computer's **instruction set**.

Each computer instruction allows the programmer to initiate the performance of a specific operation. All computers implement certain arithmetic operations in their instruction set, such as an instruction to add the contents of two registers. Often logical operations (for example, OR the contents of two registers) and register operate instructions (for example, increment a register) are included in the instruction set. A computer's instruction set also has instructions that move data between registers, between a register and memory, and between a register and an I/O device. Most instruction sets also provide **conditional instructions**. A conditional instruction specifies an operation to be performed only if certain conditions have been met; for example, jump to a particular instruction if the result of the last operation was zero. Conditional instructions provide a program with a decision-making capability.

By logically organizing a sequence of instructions into a coherent program, the programmer can "tell" the computer to perform a very specific and useful function.

The computer, however, can only execute programs whose instructions are in a binary coded form (for example, a series of 1's and 0's), that is called **machine code**. Because it would be extremely cumbersome to program in machine code, programming languages have been developed. There are programs available which convert the programming language instructions into machine code that can be interpreted by the processor.

One type of programming language is **assembly language**. A unique assembly language mnemonic is assigned to each of the computer's instructions. The programmer can write a program (called the **source program**) using these mnemonics and certain operands; the source program is then converted into machine instructions (called the **object code**). Each assembly language instruction is converted into one machine code instruction (1 or more bytes) by an **assembler** program. Assembly languages are usually machine dependent (for example, they are usually able to run on only one type of computer).

THE 8080 INSTRUCTION SET

This computer uses a Z80 microprocessor, which provides a great deal of flexibility in programming for you the user. However, HEATH has chosen to support the more popular (and more familiar to most) instruction set of the 8080A. Therefore, when you use the HEATH assemblers, the documentation shows execution times and CPU responses for the 8080A rather than the Z80.

Since some routines are time dependent, knowing the execution time for each instruction is essential. Also, when doing some sort of arithmetic operation, the conditions which affect the setting of the CPU flags must be known. To help you find the necessary information about each instruction, a cross reference from 8080A to Z80 mnemonics is shown below. From this, you can refer to the following Z80 section of the Manual.

Even though HASL-8 and ASM will not accept the Z80 mnemonics, you may use the DB and DW pseudos to take advantage of any Z80 instructions you may wish to use. An example using the DB pseudo is shown below.

```
                                1  **      EXAMPLE OF USE OF A DEFINE BYTE (DB) STATEMENT FOR USING
                                2  *       Z80 INSTRUCTIONS WHICH ARE NOT SUPPORTED BY THE HEATH
                                3  *       8080A ASSEMBLER
                                4
                                5
   000.355                      6  LDIR1   EQU     11101101B       Z80 LDIR INSTRUCTION BYTE 1
   000.260                      7  LDIR2   EQU     10110000B       Z80 LDIR INSTRUCTION BYTE 2
                                8
                                9
                               10
   000.000                     11          ORG     0
                               12
   000.000  041 000 100        13          LXI     H,BUFF1         COPY FROM BUFFER #1
   000.003  021 000 104        14          LXI     D,BUFF2         TO BUFFER #2
   000.006  001 000 004        15          LXI     B,1024          BUFFER LENGTH IS 1K
                               16
                               17  *       LDIR                    EXECUTE LDIR COPY
   000.011  355 260            18          DB      LDIR1,LDIR2
                               19
                               20
                               21
                               22  **      RAM
                               23
   100.000                     24          ORG     100000A
                               25
   100.000                     26  BUFF1   DS      1024
   104.000                     27  BUFF2   DS      1024
                               28
   110.000                     29          END
ASSEMBLY COMPLETE
   29 STATEMENTS
    0 ERRORS DETECTED
15842 BYTES FREE
```

The 8080 instruction set includes five different types of instructions:

- **Data Transfer Group** — move data between registers or between memory and registers.
- **Arithmetic Group** — add, subtract, increment, or decrement data in registers or in memory.
- **Logical Group** — AND, OR, EXCLUSIVE-OR, compare, rotate, or complement data in registers or in memory.
- **Branch Group** — conditional and unconditional jump instructions, subroutine call instructions, and return instructions.
- **Stack, I/O, and Machine Control Group** — includes I/O instructions, as well as instructions for maintaining the stack and internal control flags.

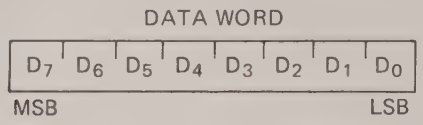
Instruction and Data Formats

Memory for the 8080 is organized into 8-bit quantities called bytes. Each byte has a unique 16-bit binary address corresponding to its sequential position in memory.

The 8080 can directly address up to 65,536 bytes of memory, which may consist of both read-only memory (ROM) elements and random-access memory (RAM) elements (read/write memory).

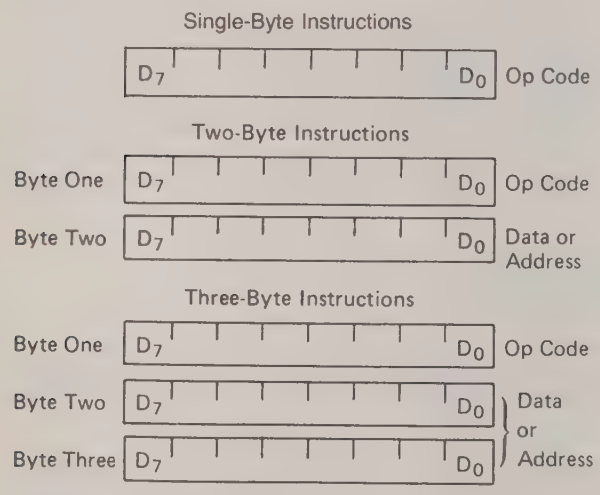
8080 Mnemonic	Z-80 Mnemonic	8080 Mnemonic	Z-80 Mnemonic	8080 Mnemonic	Z-80 Mnemonic
ACI	ADC A,N	IN	IN A,(N)	POP H	POP HL
ADC M	ADC A,(HL)	INR M	INC (HL)	POP PSW	POP AF
ADC r	ADC A,R	INR r	INC R	PUSH B	PUSH BC
ADD M	ADD A,(HL)	INX B	INC BC	PUSH D	PUSH DE
ADD r	ADD A,R	INX D	INC DE	PUSH H	PUSH HL
ADI	ADD A,N	INX H	INC HL	PUSH PSW	PUSH AF
ANA M	AND (HL)	INX SP	INC SP	RAL	RLA
ANA r	AND R	JC	JP C,NN	RAR	RRA
ANI	AND N	JM	JP M,NN	RC	RET C
CALL	CALL NN	JMP	JP NN	RET	RET
CC	CALL C,NN	JNC	JP NC,NN	RLC	RLCA
CM	CALL M,NN	JNZ	JP NZ,NN	RM	RET M
CMA	CPL	JP	JP P,NN	RNC	RET NC
CMC	CCF	JPE	JP PE,NN	RNZ	RET NZ
CMP M	CP (HL)	JPO	JP PO,NN	RP	RET P
CMP r	CP R	JZ	JP Z,NN	RPE	RET PE
CNC	CALL NC,NN	LDA	LD A,(NN)	RPO	RET PO
CNZ	CALL NZ,NN	LDAX B	LD A,(BC)	RRC	RRCA
CP	CALL P,NN	LDAX D	LD A,(DE)	RST	RST P
CP E	CALL PE,NN	LHLD	LD HL,(NN)	RZ	RET Z
CPI	CP N	LXI B	LD BC,NN	SBB M	SBC A,(HL)
CPO	CALL PO,NN	LXI D	LD DE,NN	SBB r	SBC A,R
CZ	CALL Z,NN	LXI H	LD HL,NN	SBI	SBC A,N
DAA	DAA	LXI SP	LD SP,NN	SHLD	LD (NN),HL
DAD B	ADD HL,BC	MVI M	LD (HL),N	SPHL	LD SP,HL
DAD D	ADD HL,DE	MVI r	LD R,N	STA	LD (NN),A
DAD H	ADD HL,HL	MOV M,r	LD (HL),R	STAX B	LD (BC),A
DAD SP	ADD HL,SP	MOV r,M	LD R,(HL)	STAX D	LD (DE),A
DCR M	DEC (HL)	MOV r1,r2	LD R,R'	STC	SCF
DCR r	DEC R	NOP	NOP	SUB M	SUB (HL)
DCX B	DEC BC	ORA M	OR (HL)	SUB r	SUB R
DCX D	DEC DE	ORA r	OR R	SUI	SUB N
DCX H	DEC HL	ORI	OR N	XCHG	EX DE,HL
DCX SP	DEC SP	OUT	OUT (N),A	XRA M	XOR (HL)
DI	DI	PCHL	JP (HL)	XRA r	XOR R
EI	EI	POP B	POP BC	XRI	XOR N
HLT	HALT	POP D	POP DE	XTHL	EX (SP),HL

Data in the 8080 is stored in the form of 8-bit binary integers:



When a register or data word contains a binary number, it is necessary to establish the order in which the bits of the number are written. In the 8080, BIT 0 is referred to as the **Least Significant Bit (LSB)**, and BIT 7 (of an 8-bit number) is referred to as the **Most Significant Bit (MSB).**

The 8080 program instructions may be one, two, or three bytes in length. Multiple byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instructions. The exact instruction format will depend on the particular operation to be executed.



Addressing Modes

Often the data to be operated on is stored in memory. When multi-byte numeric data is used, the data, like instructions, is stored in successive memory locations, with the least significant byte first, followed by increasingly significant bytes. The 8080 has four different modes for addressing data stored in memory or in registers:

- **Direct** — Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low-order bits of the address are in byte 2, the high-order bits in byte 3).

- **Register** — The instruction specifies the register or register pair in which the data is located.

- **Register Indirect** — The instruction specifies a register pair which contains the memory address where the data is located (the high-order bits of the address are in the first register of the pair, the low-order bits in the second).

- **Immediate** — The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

Unless directed by an interrupt or branch instruction, the execution of instructions proceeds through consecutively increasing memory locations. A branch instruction can specify the address of the next instruction to be executed in one of two ways:

- **Direct** — The branch instruction contains the address of the next instruction to be executed. (Except for the "RST" instruction, byte 2 contains the low-order address and byte 3 the high-order address.)

- **Register Indirect** — The branch instruction indicates a register pair which contains the address of the next instruction to be executed. (The high-order bits of the address are in the first register of the pair, the low-order bits in the second.)

The RST instruction is a special 1-byte call instruction (usually used during interrupt sequences). RST includes a 3-bit field; program control is transferred to the instruction whose address is eight times the contents of this 3-bit field.

Condition Flags

There are five condition flags associated with the execution of instructions on the 8080. They are Zero, Sign, Parity, Carry, and Auxiliary Carry, and are each represented by a 1-bit register in the CPU. A flag is "set" by forcing the bit to 1; "reset" by forcing the bit to 0.

Unless indicated otherwise, when an instruction affects a flag, it affects it in the following manner.

Zero: If the result of an instruction has the value 0, this flag is set; otherwise it is reset.

Sign: If the most significant bit of the result of the operation has the value 1, this flag is set; otherwise it is reset.

Parity: If the modulo 2 sum of the bits of the result of the operation is 0 (for example, if the result has even parity), this flag is set; otherwise it is reset (for example, if the result has odd parity).

Carry: If the instruction resulted in a carry (from addition), or a borrow (from subtraction or a comparison) out of the high-order bit, this flag is set; otherwise it is reset.

Auxiliary Carry: If the instruction caused a carry out of bit 3 and into bit 4 of the resulting value, the auxiliary carry is set; otherwise it is reset. This flag is affected by single precision additions, subtractions, increments, decrements, comparisons, and logical operations, but is principally used with additions and increments preceding a DAA (Decimal Adjust Accumulator) instruction.

Symbols and Abbreviations

The following symbols and abbreviations are used in the subsequent description of the 8080 instructions:

SYMBOLS	MEANING
accumulator	Register A
addr	16-bit address quantity
data	8-bit data quantity
data 16	16-bit data quantity
byte 2	The second byte of the instruction
byte 3	The third byte of the instruction
port	8-bit address of an I/O device
r, r1, r2	One of the registers A, B, C, D, E, H, L
DDD, SSS	The bit pattern designating one of the registers A, B, C, D, E, H, L (DDD = destination, SSS = source):

DDD or SSS		REGISTER NAME
BINARY	OCTAL	
111	7	A
000	0	B
001	1	C
010	2	D
011	3	E
100	4	H
101	5	L

rp One of the register pairs:

B represents the B, C pair with B as the high-order register and C as the low-order register;

D represents the D, E pair with D as the high-order register and E as the low-order register;

H represents the H, L pair with H as the high-order register and L as the low-order register;

SP represents the 16-bit stack pointer register.

RP The bit pattern designating one of the register pairs B, D, H, SP:

RP	REGISTER PAIR
00	B-C
01	D-E
10	H-L
11	SP

rh The first (high-order) register of a designated register pair.

rl The second (low-order) register of a designated register pair.

PC 16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8-bits, respectively).

SP 16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8-bits, respectively).

rm	Bit m of the register r (bits are numbered 7 through 0 from left to right).
Z, S, P, CY, AC	The condition flags: Zero, Sign, Parity, Carry, and Auxiliary-Carry, respectively.
( )	The contents of the memory location or registers enclosed in the parentheses.
←	"Is transferred to"
∧	Logical AND
∀	Exclusive OR
∨	Inclusive OR
+	Addition
–	Two's complement subtraction
*	Multiplication
↔	"Is exchanged with"
—	The one's complement (e. g., $(\overline{A})$)
n	The restart number 0 through 7
NNN	The binary representation 000 through 111 for restart number 0 through 7 respectively.

Description Format

The following pages provide a detailed description of the instruction set of the 8080. Each instruction is described in the following manner:

1. The numbers above the mnemonic are the octal opcodes for the instruction.

2. The assembler format, consisting of the instruction mnemonic and operand fields, is printed in **BOLDFACE** on the left side of the first line.

3. The name of the instruction is enclosed in parentheses on the right side of the first line.

4. The next line(s) contain a symbolic description of the operation of the instruction.

5. This is followed by a narrative description of the operation of the instruction.

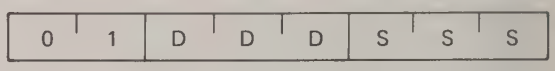
6. The following line(s) contain the binary fields and patterns that comprise the machine instruction.

7. The last four lines contain incidental information about the execution of the instruction. The number of machine cycles and states required to execute the instruction are listed first. If the instruction has two possible execution times, as in a conditional jump, both times will be listed, separated by a slash. Next, any significant data addressing modes are listed. The last line lists any of the five Flags that are affected by the execution of the instruction.

Data Transfer Group

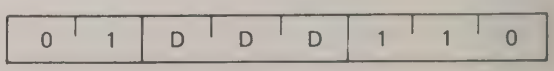
This group of instructions transfers data to and from registers and memory. **Condition flags are not affected** by any instruction in this group.

1 (0-5,7) (0-5,7)
MOV r1, r2 (Move Register)
(r1) ← (r2)
The content of register r2 is moved to register r1.



Cycles: 1
States: 5
Addressing: register
Flags: none

1(0-7)6
MOV r, M (Move from memory)
(r) ← ((H) (L))
The content of the memory location, whose address is in registers H and L, is moved to register r.



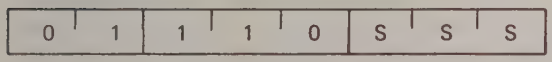
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

16 (0-7)

MOV M, r (Move to memory)

((H) (L)) ← (r)

The content of register r is moved to the memory location whose address is in registers H and L.



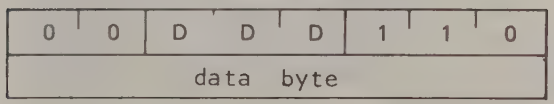
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

0 (0-7)6

MVI r, data (Move Immediate) 0(0-7)6

(r) ← (byte 2)

The content of byte 2 of the instruction is moved to register r.



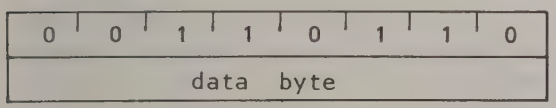
Cycles: 2
States: 7
Addressing: immediate
Flags: none

066

MVI M, data (Move to memory immediate)

((H) (L)) ← (byte 2)

The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.



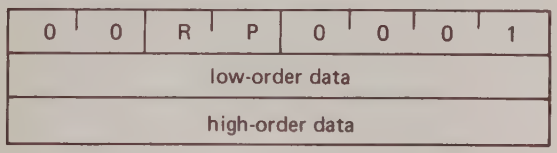
Cycles: 3
States: 10
Addressing: immed./reg. indirect
Flags: none

001 (B, C) 041 (H, L)
021 (D, E) 061 (S, P)

LXI rp, data 16 (Load register pair immediate)

(rh) ← (byte 3),
(rl) ← (byte 2)

Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.



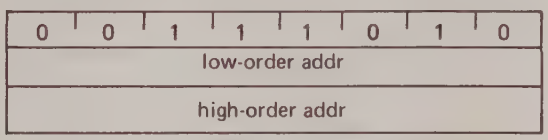
Cycles: 3
States: 10
Addressing: immediate
Flags: none

072

LDA addr (Load Accumulator direct)

(A) ← ((byte 3) (byte 2))

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to the accumulator.



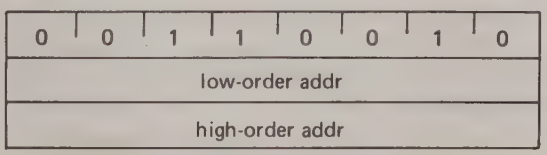
Cycles: 4
States: 13
Addressing: direct
Flags: none

062

STA addr (Store Accumulator direct)

((byte 3) (byte 2)) ← (A)

The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.



Cycles: 4
States: 13
Addressing: direct
Flags: none

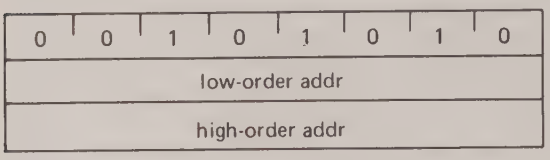
052

LHLD addr (Load H and L direct)

(L) ← ((byte 3) (byte 2))

(H) ← ((byte 3) (byte 2) + 1)

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register L. The content of the memory location at the succeeding address is moved to register H.



Cycles: 5
States: 16
Addressing: direct
Flags: none

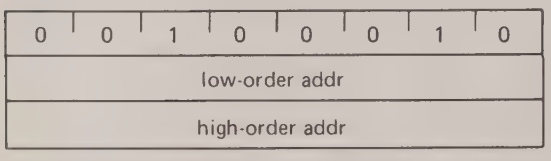
042

SHLD addr (Store H and L direct)

((byte 3) (byte 2)) ← (L)

((byte 3) (byte 2) + 1) ← (H)

The content of register L is moved to the memory location whose address is specified in byte 2 and byte 3. The content of register H is moved to the succeeding memory location.



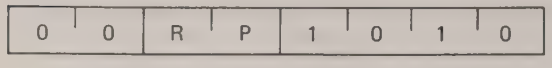
Cycles: 5
States: 16
Addressing: direct
Flags: none

012 (B, C) 032 (D, E)

LDAX rp (Load accumulator indirect)

(A) ← ((rp))

The content of the memory location, whose address is in the register pair rp, is moved to register A. NOTE: Only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

002 (B, C) 022 (D, E)

STAX rp (Store accumulator indirect)

((rp)) ← (A)

The content of register A is moved to the memory location whose address is in the register pair rp. NOTE: Only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

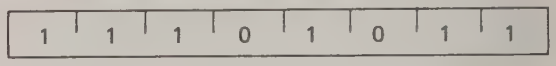
353

XCHG (Exchange H and L with D and E)

(H) ↔(D)

(L) ↔(E)

The contents of registers H and L are exchanged with the contents of registers D and E.



Cycles: 1
States: 4
Addressing: register
Flags: none

Arithmetic Group

This group of instructions performs arithmetic operations on data in registers and memory.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Carry, and Auxiliary Carry flags according to the standard rules.

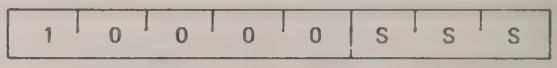
All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

20 (0-5,7)

ADD r (Add Register)

(A) ← (A) + (r)

The content of register r is added to the content of the accumulator. The result is placed in the accumulator.

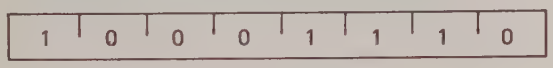


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

206
ADD M (Add memory)
(A) ← (A) + ((H) (L))
The content of the memory location whose address is contained in the H and L registers is added to the content of the accumulator. The result is placed in the accumulator.

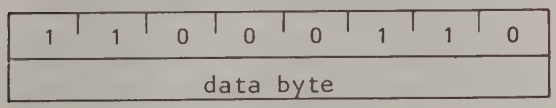
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

216
ADC M (Add memory with carry)
(A) ← (A) + ((H) (L)) + (CY)
The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are added to the content of the accumulator. The result is placed in the accumulator.



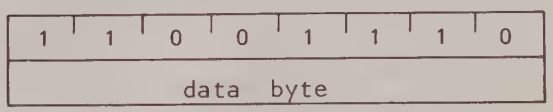
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

306
ADI DATA (Add immediate)
(A) ← (A) + (byte 2)
The content of the second byte of the instruction is added to the content of the accumulator. The result is placed in the accumulator.



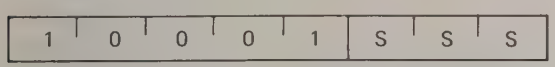
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

316
ACI data (Add immediate with carry)
(A) ← (A) + (byte 2) + (CY)
The content of the second byte of the instruction and the content of the CY flag are added to the content of the accumulator. The result is placed in the accumulator.



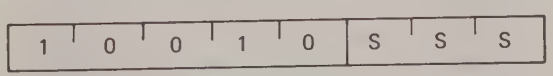
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

21 (0-5,7)
ADC r (Add Register with carry)
(A) ← (A) + (r) + (CY)
The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.



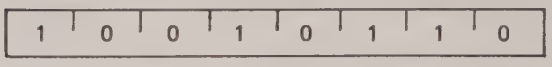
Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

22 (0-5,7)
SUB r (Subtract Register)
(A) ← (A) − (r)
The content of register r is subtracted from the content of the accumulator. The result is placed in the accumulator.



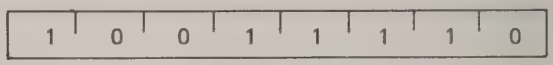
Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

226
SUB M (Subtract memory)
(A) ← (A) − ((H) (L))
The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.



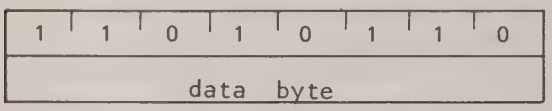
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

236
SBB M (Subtract memory with borrow)
(A) ← (A) − ((H) (L)) − (CY)
The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the content of the accumulator. The result is placed in the accumulator.



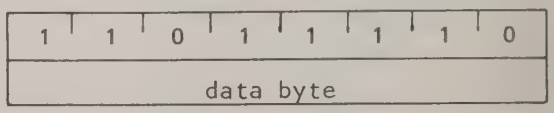
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

326
SUI data (Subtract immediate)
(A) ← (A) − (byte 2)
The content of the second byte of the instruction is subtracted from the content of the accumulator. The result is placed in the accumulator.



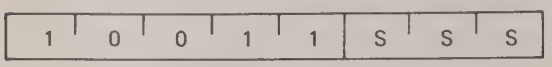
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

336
SBI data (Subtract immediate with borrow)
(A) ← (A) − (byte 2) − (CY)
The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the content of the accumulator. The result is placed in the accumulator.



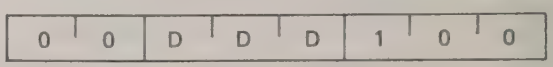
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

23 (0-5,7)
SBB r (Subtract Register with borrow)
(A) ← (A) − (r) − (CY)
The content of register r and the content of the CY flag are both subtracted from the content of the accumulator. The result is placed in the accumulator.



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

0 (0-5,7)4
INR r (Increment Register)
(r) ← (r) + 1
The content of register r is incremented by one.
NOTE: All condition flags **except CY** are affected.



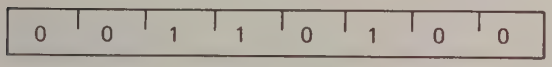
Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

064

INR M (Increment memory)

((H) (L)) ← ((H) (L)) + 1

The content of the memory location whose address is contained in the H and L registers is incremented by one. NOTE: All condition flags **except CY** are affected.



Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

0 (0-5,7)5

DCR r (Decrement Register)

(r) ← (r) − 1

The content of register r is decremented by one. NOTE: All condition flags **except CY** are affected.

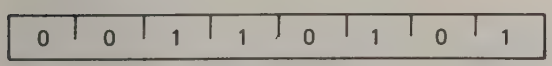
Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

065

DCR M (Decrement memory)

((H) (L)) ← ((H) (L)) − 1

The content of the memory location whose address is contained in the H and L registers is decremented by one. NOTE: All condition flags **except CY** are affected.



Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

003 (B,C) 043 (H,L)
023 (D,E) 063 (S,P)

INX rp (Increment register pair)

(rh) (rl) ← (rh) (rl) + 1

The content of the register pair rp is incremented by one. NOTE: **No condition flags are affected.**

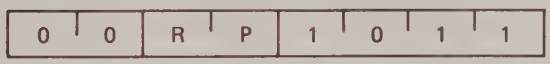
Cycles: 1
States: 5
Addressing: register
Flags: none

013 (B,C) 053 (H,L)
033 (D,E) 073 (S,P)

DCX rp (Decrement register pair)

(rh) (rl) ← (rh) (rl) − 1

The content of the register pair rp is decremented by one. NOTE: **No condition flags are affected.**



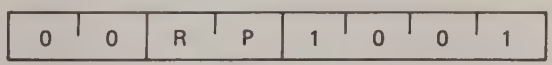
Cycles: 1
States: 5
Addressing: register
Flags: none

011 (B,C) 051 (H,L)
031 (D,E) 071 (S,P)

DAD rp (Add register pair to H and L)

(H) (L) ← (H) (L) + (rh) (rl)

The content of the register pair rp is added to the content of the register pair H and L. The result is placed in the register pair H and L. NOTE: **Only the CY flag is affected.** It is set if there is a carry out of the double precision add; otherwise it is reset.



Cycles: 3
States: 10
Addressing: register
Flags: CY

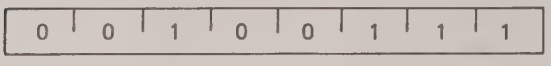
047
DAA (Decimal Adjust Accumulator)

The eight-bit number in the accumulator is adjusted to form two 4-bit Binary-Coded-Decimal digits by the following process:

1. If the value of the least significant 4 bits of the accumulator is greater than 9, **or** if the AC flag is set, 6 is added to the accumulator.

2. If the value of the most significant 4 bits of the accumulator is now greater than 9, **or** if the CY flag is set, 6 is added to the most significant 4 bits of the accumulator.

NOTE: All flags are affected.



Cycles: 1
States: 4
Flags: Z,S,P,CY,AC

Logical Group:

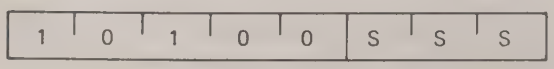
This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and Carry flags according to the standard rules.

24 (0-5,7)
ANA r (AND Register)

(A) ← (A) ∧ (r)

The content of register r is logically anded with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared.**

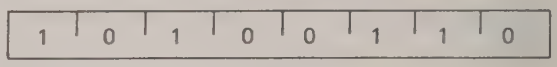


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

246
ANA M (AND memory)

(A) ← (A) ∧ ((H) (L))

The contents of the memory location whose address is contained in the H and L registers is logically anded with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared.**

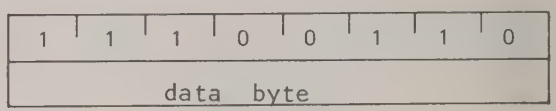


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

346
ANI data (AND immediate)

(A) ← (A) ∧ (byte 2)

The content of the second byte of the instruction is logically anded with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

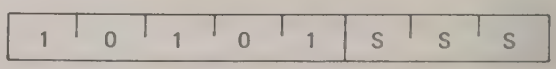


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

25 (0-5,7)
XRA r (Exclusive OR Register)

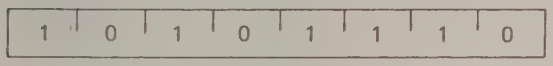
(A) ← (A) ∀ (r)

The content of register r is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

256
XRA M (Exclusive OR Memory)
(A) ← (A) ⊻ ((H) (L))
The content of the memory location whose address is contained in the H and L registers is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

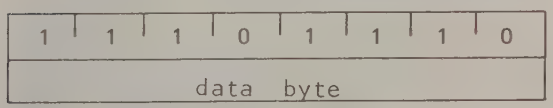


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

266
ORA M (OR memory)
(A) ← (A) ∨ ((H) (L))
The content of the memory location whose address is contained in the H and L registers is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

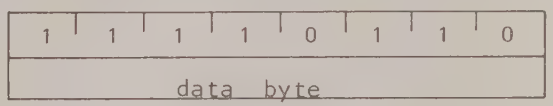
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

356
XRI data (Exclusive OR immediate)
(A) ← (A) ⊻ (byte 2)
The content of the second byte of the instruction is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**



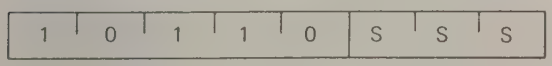
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

366
ORI data (OR Immediate)
(A) ← (A) ∨ (byte 2)
The content of the second byte of the instruction is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**



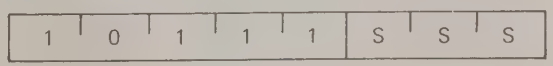
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

26 (0-5,7)
ORA r (OR Register)
(A) ← (A) ∨ (r)
The content of register r is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

27 (0-5,7)
CMP r (Compare Register)
(A) − (r)
The content of register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if (A) = (r). The CY flag is set to 1 if (A) < (r).**



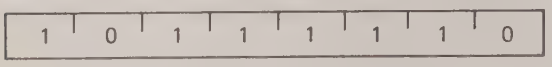
Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

276

CMP M (Compare memory)

(A) − ((H) (L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if (A) = ((H) (L)). The CY flag is set to 1 if (A) < ((H) (L)).**



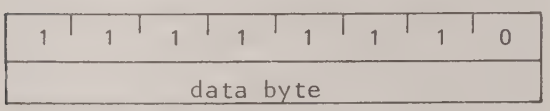
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

376

CPI data (Compare immediate)

(A) − (byte 2)

The content of the second byte of the instruction is subtracted from the content of the accumulator. The condition flags are set by the result of the subtraction. **The Z flag is set to 1 if (A) = (byte 2). The CY flag is set to 1 if (A)** < (byte 2).



Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

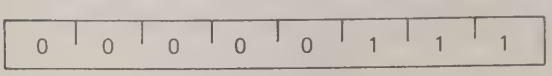
007

RLC (Rotate left)

$(A_{n+1}) \leftarrow (A_n)$; $(A_0) \leftarrow (A_7)$

$(CY) \leftarrow (A_7)$

The content of the accumulator is rotated left one position. The low-order bit and the CY flag are both set to the value shifted out of the high-order bit position. **Only the CY flag is affected.**



Cycles: 1
States: 4
Flags: CY

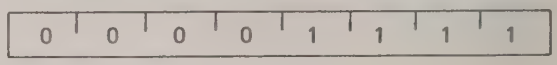
017

RRC (Rotate right)

$(A_n) \leftarrow (A_{n-1})$; $(A_7) \leftarrow (A_0)$

$(CY) \leftarrow (A_0)$

The content of the accumulator is rotated right one position. The high-order bit and the CY flag are both set to the value shifted out of the low-order bit position. **Only the CY flag is affected.**



Cycles: 1
States: 4
Flags: CY

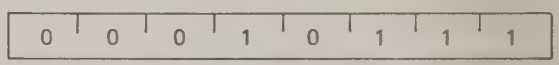
027

RAL (Rotate left through carry)

$(A_{n+1}) \leftarrow (A_n)$; $(CY) \leftarrow (A_7)$

$(A_0) \leftarrow (CY)$

The content of the accumulator is rotated left one position through the CY flag. The low-order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high-order bit. **Only the CY flag is affected.**



Cycles: 1
States: 4
Flags: CY

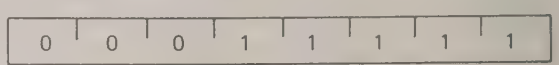
037

RAR (Rotate right through carry)

$(A_n) \leftarrow (A_{n+1})$; $(CY) \leftarrow (A_0)$

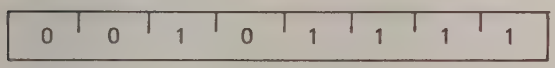
$(A_7) \leftarrow (CY)$

The content of the accumulator is rotated right one position through the CY flag. The high-order bit is set to the CY flag and the CY flag is set to the value shifted out of the low-order bit. **Only the CY flag is affected.**



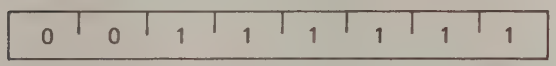
Cycles: 1
States: 4
Flags: CY

057
CMA (Complement accumulator)
(A) ← (A)
The content of the accumulator is complemented (zero bits become 1, one bits become 0). **No flags are affected.**



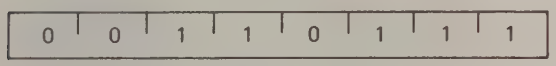
Cycles: 1
States: 4
Flags: none

077
CMC (Complement carry)
(CY) ← ($\overline{CY}$)
The CY flag is complemented. **No other flags are affected**.



Cycles: 1
States: 4
Flags: CY

067
STC (Set carry)
(CY) ← 1
The CY flag is set to 1. **No other flags are affected**.



Cycles: 1
States: 4
Flags: CY

Branch Group

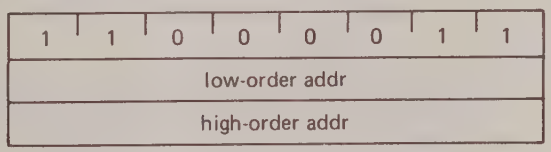
This group of instructions alters normal sequential program flow.

Condition flags are not affected by any instruction in this group.

The two types of branch instructions are **unconditional** and **conditional**. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

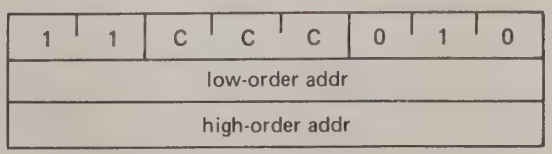
CONDITION	CCC	OCTAL
NZ — not zero (Z = 0)	000	0
Z — zero (Z = 1)	001	1
NC — no carry (CY = 0)	010	2
C — carry (CY = 1)	011	3
PO — parity odd (P = 0)	100	4
PE — parity even (P = 1)	101	5
P — plus (S = 0)	110	6
M — minus (S = 1)	111	7

303
JMP addr (Jump)
(PC) ← (byte 3) (byte 2)
Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



Cycles: 3
States: 10
Addressing: immediate
Flags: none

3 (0-7)2
Jcondition addr (Condition jump)
If (CCC),
(PC) ← (byte 3) (byte 2)
If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.



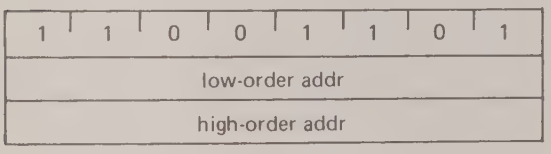
Cycles: 3
States: 10
Addressing: immediate
Flags: none

315

CALL addr (Call)

((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← (byte 3) (byte 2)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



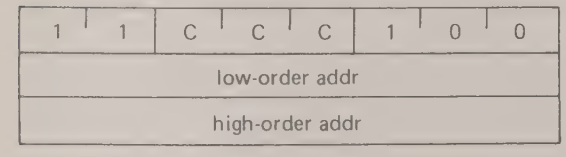
Cycles: 5
States: 17
Addressing: immediate/reg. indirect
Flags: none

3 (0-7) 4

Ccondition addr (Condition call)

If (CCC),
((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← (byte 3) (byte 2)

If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.



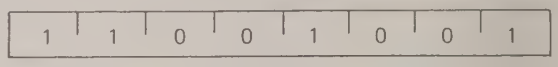
Cycles: 3/5
States: 11/17
Addressing: immediate/reg. indirect
Flags: none

311

RET (Return)

(PCL) ← ((SP));
(PCH) ← ((SP) + 1);
(SP) ← (SP) + 2;

The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose address is one more than the content of register SP is moved to the high-order eight bits of register PC. The content of register SP is incremented by 2.



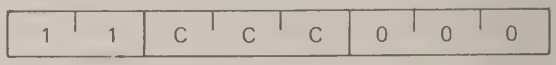
Cycles: 3
States: 10
Addressing: reg. indirect
Flags: none

3 (0-7) 0

Rcondition (Conditional return)

If (CCC),
(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
(SP) ← (SP) + 2

If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.



Cycles: 1/3
States: 5/11
Addressing: reg. indirect
Flags: none

3 (0-7)7

RST n (Restart)

((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← 8 * (NNN)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction

address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.

1	1	N	N	N	1	1	1

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	N	N	N	0	0	0

Program Counter After Restart

351

PCHL (Jump H and L indirect — move H and L to PC)

(PCH) ← (H)
(PCL) ← (L)
The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.

1	1	1	0	1	0	0	1

Cycles: 1
States: 5
Addressing: register
Flags: none

Stack, I/O, and Machine Control Group

This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, **condition flags are not affected by any instructions in this group.**

305 (B, C) 345 (H, L)
325 (D, E)

PUSH rp (Push)

((SP) − 1) ← (rh)
((SP) − 2) ← (rl)
(SP) ← (SP) − 2
The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. **NOTE: Register pair rp = SP may not be specified.**

1	1	R	P	0	1	0	1

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

365

PUSH PSW (Push processor status word)

((SP) − 1) ← (A)
((SP) − 2)$_0$ ← (CY), ((SP) − 2)$_1$ ← 1
((SP) − 2)$_2$ ← (P), ((SP) − 2)$_3$ ← 0
((SP) − 2)$_4$ ← (AC), ((SP) − 2)$_5$ ← 0
((SP) − 2)$_6$ ← (Z), ((SP) − 2)$_7$ ← (S)
(SP) ← (SP) − 2
The content of the accumulator is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2.

1	1	1	1	0	1	0	1

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

FLAG WORD

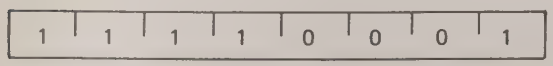
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	0	AC	0	P	1	CY

301 (B,C)
321 (D,E)
341 (H,L)

POP rp (Pop)
(rl) ← ((SP))
(rh) ← ((SP) + 1)
(SP) ← (SP) + 2
The content of the memory location, whose address is specified by the content of register SP is moved to the low-order register of register pair rp. The content of the memory location whose address is one more than the content of register SP is moved to the high-order register of register pair rp. The content of register SP is incremented by 2. **NOTE: Register pair rp = SP may not be specified.**

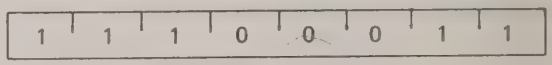
Cycles: 3
States: 10
Addressing: reg. indirect
Flags: none

361
POP PSW (Pop processor status word)
(CY) ← $((SP))_0$
(P) ← $((SP))_2$
(AC) ← $((SP))_4$
(Z) ← $((SP))_6$
(S) ← $((SP))_7$
(A) ← ((SP) + 1)
(SP) ← (SP) + 2
The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.



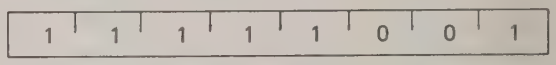
Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

343
XTHL (Exchange stack top with H and L)
(L) ↔ ((SP))
(H) ↔ ((SP) + 1)
The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.



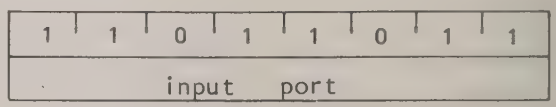
Cycles: 5
States: 18
Addressing: reg. indirect
Flags: none

371
SPHL (Move HL to SP)
(SP) ← (H) (L)
The contents of registers H and L (16 bits) are moved to register SP.



Cycles: 1
States: 5
Addressing: register
Flags: none

333
IN port (Input)
(A)←(data)
The data placed on the eight-bit bi-directional data bus by the specified port is moved to the accumulator.



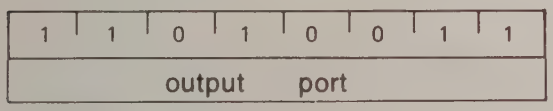
Cycles: 3
States: 10
Addressing: direct
Flags: none

323

OUT port (Output)

(data) ← (A)

The content of the accumulator is placed on the eight-bit bi-directional data bus for transmission to the specified port.



Cycles: 3
States: 10
Addressing: direct
Flags: none

373

EI (Enable interrupt)

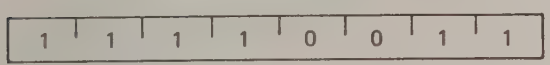
The interrupt system is enabled **following the execution of the next instruction**.

Cycles: 1
States: 4
Flags: none

363

DI (Disable interrupt)

The interrupt system is disabled **immediately following the execution of the DI instruction**.



Cycles: 1
States: 4
Flags: none

166

HLT (Halt)

The processor is stopped. The registers and flags are unaffected.

Cycles: 1
States: 7
Flags: none

000

NOP (No op)

No operation is performed. The registers and flags are unaffected.

Cycles: 1
States: 4
Flags: none

DEMONSTRATION PROGRAMS

These BASIC programs demonstrate some of the Computer features. These include:

- Erase Page
- Direct Cursor Addressing
- Graphics
- Reverse Video
- 25th Line

If you are using the HDOS, you must first map for lower case output before the programs will work. To do this, first boot up your system. Then type:

```
>SET TT: NOMLO
```

Since there are differences between BASIC languages, you may have to change the syntax of these programs slightly to get them to run on a non-Heath computer system.

DEMONSTRATION PROGRAM #1

This program draws a simple maze on the screen.

NOTE: Notice the semicolon at the end of line 350. This prevents a Carriage Return and a Line Feed, and the cursor remains at its present location on the line. Normally this is acceptable. However, some BASIC languages count the number of characters sent to the Terminal and automatically insert their own Carriage Return and Line Feed. If this automatic CR & LF comes during a sucessive execution of line 350, the direct cursor addressing sequence is upset and the character is placed randomly on the screen. To prevent this, a PRINT statement has been placed at line 440. This forces a CR & LF every ninth execution of line 350, thus preventing an automatic CR & LF.

The semicolon at the end of line 350 and the PRINT statement at line 440 can both be eliminated. However, the cursor will return to the left side of the screen after each execution of line 350. This is also acceptable, but the cursor will jump back and forth between the left side and the center of the screen.

```
00010 REM  Demonstration Program #1
00020 REM
00030 DIM R(8,18),Q$(6)
00040 REM Read Data
00050 FOR I=1 TO 6
00060 READ Q$(I)
00070 NEXT I
00080 REM Erase Page
```

```
00090 PRINT CHR$(27);CHR$(69)
00100 REM Print Message
00110 FOR I=1 TO 3
00120 PRINT Q$(I)
00130 NEXT I
00140 REM Print message on the bottom line
00150 PRINT CHR$(27);CHR$(89);CHR$(53);CHR$(43);Q$(6)
00160 REM Read Data
00170 FOR I=0 TO 8
00180 FOR J=0 TO 18
00190 READ R(I,J)
00200 NEXT J
00210 NEXT I
00220 REM Erase Bottom Line
00230 PRINT CHR$(27);CHR$(89);CHR$(53);CHR$(33);CHR$(27);CHR$(75)
00240 REM Enter Reverse Video Mode
00250 PRINT CHR$(27);CHR$(112);
00260 REM Print 'Start'
00270 PRINT CHR$(27);CHR$(89);CHR$(38);CHR$(46);Q$(4)
00280 REM Exit Reverse Video Mode
00290 PRINT CHR$(27);CHR$(113);
00300 REM Enter Graphics Mode
00310 PRINT CHR$(27);CHR$(70);
00320 I=5
00330 J=11
00340 REM Use Direct Cursor Addressing & Print 1 Graphic Character
00350 PRINT CHR$(27);CHR$(89);CHR$(41+I);CHR$(46+J);CHR$(R(I,J));
00360 REM Randomly change the values of I & J
00370 I=I+5
00380 IF I<9 THEN 400
00390 I=I-9
00400 J=J+13
00410 IF J<19 THEN 430
00420 J=J-19
00430 IF I<>5 THEN 350
00440 PRINT
00450 IF J<>11 THEN 350
00460 REM Exit Graphics Mode
```

```
00470 PRINT CHR$(27);CHR$(71);
00480 REM Enter Reverse Video Mode
00490 PRINT CHR$(27);CHR$(112);
00500 REM Print 'Finish'
00510 PRINT CHR$(27);CHR$(89);CHR$(52);CHR$(59);Q$(5)
00520 REM Exit Reverse Video Mode
00530 PRINT CHR$(27);CHR$(113);
00540 REM Move the cursor to the bottom line
00550 PRINT CHR$(27);CHR$(89);CHR$(89);CHR$(33)
00560 END
00570 DATA "This program demonstrates the 'Erase Page', 'Graphics',"
00580 DATA "'Erase To End Of Line', 'Reverse Video', and the 'Direct Cursor"
00590 DATA "Addressing' features of the HEATHKIT Video Terminal."
00600 DATA "Start"
00610 DATA "Finish"
00620 DATA "Hang on while I read the data list"
00630 DATA 102,97,100,101,115,97,115,97,97
00640 DATA 97,97,97,115,97,97,97,97,97,99
00650 DATA 118,32,116,102,100,118,115,100
00660 DATA 101,97,99,101,97,97,97,117,99,32,96
00670 DATA 96,32,117,97,115,100,118,98,99
00680 DATA 96,118,97,117,97,116,96,101,32,96
00690 DATA 118,97,97,100,118,99,101,99,101
00700 DATA 97,100,118,97,32,101,97,98,32,116
00710 DATA 96,102,100,101,100,96,96,101,115
00720 DATA 32,118,100,101,99,118,99,101,97,116
00730 DATA 96,101,99,96,32,96,96,118,97
00740 DATA 97,98,97,100,118,97,117,99,32,96
00750 DATA 96,32,96,101,115,97,117,99,118
00760 DATA 32,98,99,32,100,102,99,96,32,96
00770 DATA 96,32,101,97,116,101,115,97,98
00780 DATA 32,96,101,98,97,100,96,101,97,116
00790 DATA 101,97,97,97,117,97,97,117,97
00800 DATA 97,97,97,117,97,97,99,102,97,100
```

DEMONSTRATION PROGRAM #2

This program demonstrates the "25th line" and the "remember the cursor position" features.

```
00010 REM "25th Line Demo Program"
00020 REM Erase Page
00030 PRINT CHR$(27); CHR$(69)
00040 PRINT "This program demonstrates the 'twenty-fifth line' feature."
00050 PRINT "In this demonstration, the 25th line is being used as a label"
00060 PRINT "for the top row of special function keys. Reverse video"
00070 PRINT "is used to make the labels stand out better, and also to"
00080 PRINT "help avoid confusion with any normal text on the screen above"
00090 PRINT "this line. You may now run another program. Line 25 will stay"
00100 PRINT "as it is until it is changed or until this unit is RESET or"
00110 PRINT "turned off."
00120 REM Remember The Cursor Position
00130 PRINT CHR$(27); CHR$(106)
00140 REM Enable 25th Line
00150 PRINT CHR$(27); CHR$(120); CHR$(49)
00160 REM Position Cursor At Start Of 25th Line
00170 PRINT CHR$(27); CHR$(89); CHR$(56); CHR$(32)
00180 PRINT
00190 REM Enter Reverse Video Mode
00200 PRINT CHR$(27); CHR$(112);
00210 REM Print 25th Line
00220 PRINT "LINE   f1     f2     f3     f4     f5   ERASE";
00230 PRINT "  BLU    RED    GRV  RESET  BREAK";
00240 REM Exit Reverse Video
00250 PRINT CHR$(27); CHR$(113)
00260 REM Set Cursor To Previously Saved Position
00270 PRINT CHR$(27); CHR$(107)
00280 PRINT : PRINT
00290 PRINT "These lines demonstrate the 'remember cursor position' feature."
00300 PRINT "First, the above paragraph was printed; next, the 25th line"
00310 PRINT "was printed; and then these lines were printed by remembering"
00320 PRINT "the proper cursor position."
00330 PRINT :PRINT
00340 END
```

DEMONSTRATION PROGRAM #3

This program draws a reasonable facsimile of the American flag.

```
100 REM American Flag Program
110 PRINT CHR$(27);CHR$(120);CHR$(53);
120 S1$ = "* * * * * * * * * "
130 S2$ = " * * * * * * * * *"
140 E$=CHR$(155)
150 R1$ = E$ + "F"
160 R2$ = E$ + "G"
170 G1$ = E$ + "F"
180 G2$ = E$ + "G"
190 B$ = G1$ + "^" + G2$
200 P1$ = "          "+R1$+" "+R2$
210 P2$ = P1$ + G1$ + "*" + G2$
220 FOR I = 1 TO 45
230 L1$ = L1$ + "i"
240 L2$ = L2$ + " "
250 NEXT I
260 PRINT "          "+B$
270 PRINT "          "+R1$+" "+B$+R2$
280 FOR I = 1 TO 7
290 PRINT P2$;
300 IF I-2*(INT(I/2)) <> 0 THEN 320
310 PRINT S2$+R1$+MID$(L2$,1,45-LEN(S2$))+R2$:GOTO 330
320 PRINT S1$+G1$+MID$(L1$,1,45-LEN(S1$))+G2$
330 NEXT I
340 FOR I=8 TO 13
350 PRINT P2$;
360 IF I-2*(INT(I/2)) <> 0 THEN 380
370 PRINT R1$+L2$+R2$:GOTO 390
380 PRINT G1$+L1$+G2$
390 NEXT I
400 PRINT "          "R1$+" "+B$+R2$
410 PRINT P2$
420 PRINT P1$+G1$+"x"+G2$
430 PRINT P1$+G1$+"x"+G2$
440 FOR I=1 TO 4
450 PRINT P1$
460 NEXT I
470 PRINT "     "+G1$+"yyyy"+R1$+" "+R2$+"xxxx"+G2$;
480 LINE INPUT "";Z$
490 PRINT CHR$(27);"x5";
500 END
```

Z80™ CPU

The following pages are reprinted by permission of Zilog, Inc.

Copyright© 1977 by Zilog, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Zilog.

Zilog assumes no responsibility for the use of any circuitry other than circuitry embodied in a Zilog product. No other circuit patent licenses are implied.

TM: Z80 is a trademark of Zilog, Inc.

TABLE OF CONTENTS

Chapter		Page
1.0	Introduction	13-3
2.0	Z80-CPU Architecture	13-5
3.0	Z80-CPU Pin Description	13-9
4.0	CPU Timing	13-13
5.0	Z80-CPU Instruction Set	13-20
6.0	Flags	13-41
7.0	Summary of OP Codes and Execution Times	13-45
8.0	Interrupt Response	13-57
9.0	Hardware Implementation Examples	13-61
10.0	Software Implementation Examples	13-65
11.0	Electrical Specifications	13-71
12.0	Z80-CPU Instruction Set Summary	13-75

1.0 INTRODUCTION

The term "microcomputer" has been used to describe virtually every type of small computing device designed within the last few years. This term has been applied to everything from simple "microprogrammed" controllers constructed out of TTL MSI up to low end minicomputers with a portion of the CPU constructed out of TTL LSI "bit slices." However, the major impact of the LSI technology within the last few years has been with MOS LSI. With this technology, it is possible to fabricate complete and very powerful computer systems with only a few MOS LSI components.

The Zilog Z-80 family of components is a significant advancement in the state-of-the art of microcomputers. These components can be configured with any type of standard semiconductor memory to generate computer systems with an extremely wide range of capabilities. For example, as few as two LSI circuits and three standard TTL MSI packages can be combined to form a simple controller. With additional memory and I/O devices a computer can be constructed with capabilities that only a minicomputer could previously deliver. This wide range of computational power allows standard modules to be constructed by a user that can satisfy the requirements of an extremely wide range of applications.

The major reason for MOS LSI domination of the microcomputer market is the low cost of these few LSI components. For example, MOS LSI microcomputers have already replaced TTL logic in such applications as terminal controllers, peripheral device controllers, traffic signal controllers, point of sale terminals, intelligent terminals and test systems. In fact the MOS LSI microcomputer is finding its way into almost every product that now uses electronics and it is even replacing many mechanical systems such as weight scales and automobile controls.

The MOS LSI microcomputer market is already well established and new products using them are being developed at an extraordinary rate. The Zilog Z-80 component set has been designed to fit into this market through the following factors:

1. The Z-80 is fully software compatible with the popular 8080A CPU offered from several sources. Existing designs can be easily converted to include the Z-80 as a superior alternative.
2. The Z-80 component set is superior in both software and hardware capabilities to any other microcomputer system on the market. These capabilities provide the user with significantly lower hardware and software development costs while also allowing him to offer additional features in his system.
3. For increased throughput the Z80A operating at a 4 MHZ clock rate offers the user significant speed advantages over competitive products.
4. A complete product line including full software support with strong emphasis on high level languages and a disk-based development system with advanced real-time debug capabilities is offered to enable the user to easily develop new products.

Microcomputer systems are extremely simple to construct using Z-80 components. Any such system consists of three parts:

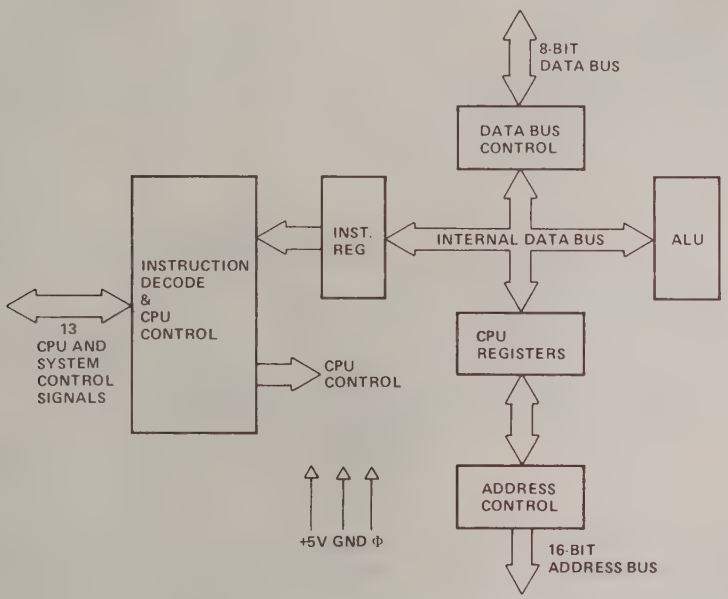
1. CPU (Central Processing Unit)
2. Memory
3. Interface Circuits to peripheral devices

The CPU is the heart of the system. Its function is to obtain instructions from the memory and perform the desired operations. The memory is used to contain instructions and in most cases data that is to be processed. For example, a typical instruction sequence may be to read data from a specific peripheral device, store it in a location in memory, check the parity and write it out to another peripheral device. Note that the Zilog component set includes the CPU and various general purpose I/O device controllers, while a wide range of memory devices may be used from any source. Thus, all required components can be connected together in a very simple manner with virtually no other external logic. The user's effort then becomes primarily one of software development. That is, the user can concentrate on describing his problem and translating it into a series of instructions that can be loaded into the microcomputer memory. Zilog is dedicated to making this step of software generation as simple as possible. A good example of this is our

assembly language in which a simple mnemonic is used to represent every instruction that the CPU can perform. This language is self documenting in such a way that from the mnemonic the user can understand exactly what the instruction is doing without constantly checking back to a complex cross listing.

2.0 Z-80 CPU ARCHITECTURE

A block diagram of the internal architecture of the Z-80 CPU is shown in figure 2.0-1. The diagram shows all of the major elements in the CPU and it should be referred to throughout the following description.



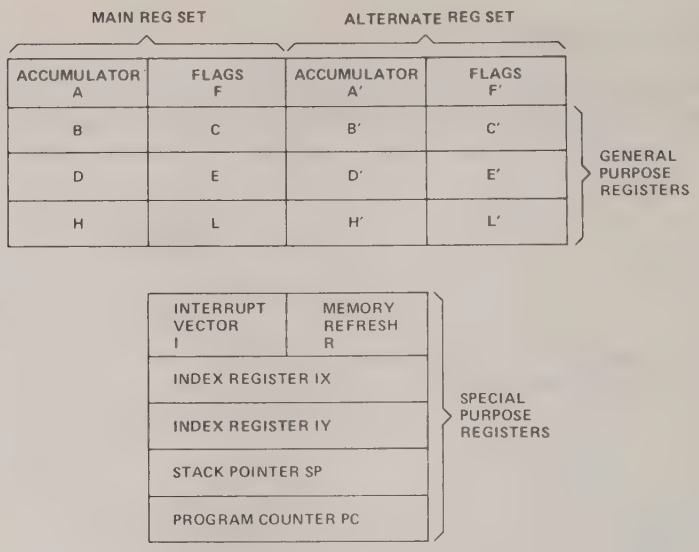
Z-80 CPU BLOCK DIAGRAM
FIGURE 2.0-1

2.1 CPU REGISTERS

The Z-80 CPU contains 208 bits of R/W memory that are accessible to the programmer. Figure 2.0-2 illustrates how this memory is configured into eighteen 8-bit registers and four 16-bit registers. All Z-80 registers are implemented using static RAM. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or in pairs as 16-bit registers. There are also two sets of accumulator and flag registers.

Special Purpose Registers

1. **Program Counter (PC).** The program counter holds the 16-bit address of the current instruction being fetched from memory. The PC is automatically incremented after its contents have been transferred to the address lines. When a program jump occurs the new value is automatically placed in the PC, overriding the incrementer.
2. **Stack Pointer (SP).** The stack pointer holds the 16-bit address of the current top of a stack located anywhere in external system RAM memory. The external stack memory is organized as a last-in first-out (LIFO) file. Data can be pushed onto the stack from specific CPU registers or popped off of the stack into specific CPU registers through the execution of PUSH and POP instructions. The data popped from the stack is always the last data pushed onto it. The stack allows simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data manipulation.



Z-80 CPU REGISTER CONFIGURATION
FIGURE 2.0-2

3. **Two Index Registers (IX & IY).** The two independent index registers hold a 16-bit base address that is used in indexed addressing modes. In this mode, an index register is used as a base to point to a region in memory from which data is to be stored or retrieved. An additional byte is included in indexed instructions to specify a displacement from this base. This displacement is specified as a two's complement signed integer. This mode of addressing greatly simplifies many types of programs, especially where tables of data are used.

4. **Interrupt Page Address Register (I).** The Z-80 CPU can be operated in a mode where an indirect call to any memory location can be achieved in response to an interrupt. The I Register is used for this purpose to store the high order 8-bits of the indirect address while the interrupting device provides the lower 8-bits of the address. This feature allows interrupt routines to be dynamically located anywhere in memory with absolute minimal access time to the routine.

5. **Memory Refresh Register (R).** The Z-80 CPU contains a memory refresh counter to enable dynamic memories to be used with the same ease as static memories. Seven bits of this 8 bit register are automatically incremented after each instruction fetch. The eighth bit will remain as programmed as the result of an LD R, A instruction. The data in the refresh counter is sent out on the lower portion of the address bus along with a refresh control signal while the CPU is decoding and executing the fetched instruction. This mode of refresh is totally transparent to the programmer and does not slow down the CPU operation. The programmer can load the R register for testing purposes, but this register is normally not used by the programmer. During refresh, the contents of the I register are placed on the upper 8 bits of the address bus.

Accumulator and Flag Registers

The CPU includes two independent 8-bit accumulators and associated 8-bit flag registers. The accumulator holds the results of 8-bit arithmetic or logical operations while the flag register indicates specific conditions for 8 or 16-bit operations, such as indicating whether or not the result of an operation is equal to zero. The programmer selects the accumulator and flag pair that he wishes to work with with a single exchange instruction so that he may easily work with either pair.

General Purpose Registers

There are two matched sets of general purpose registers, each set containing six 8-bit registers that may be used individually as 8-bit registers or as 16-bit register pairs by the programmer. One set is called BC, DE and HL while the complementary set is called BC′, DE′ and HL′. At any one time the programmer can select either set of registers to work with through a single exchange command for the entire set. In systems where fast interrupt response is required, one set of general purpose registers and an accumulator/flag register may be reserved for handling this very fast routine. Only a simple exchange commands need be executed to go between the routines. This greatly reduces interrupt service time by eliminating the requirement for saving and retrieving register contents in the external stack during interrupt or subroutine processing. These general purpose registers are used for a wide range of applications by the programmer. They also simplify programming, especially in ROM based systems where little external read/write memory is available.

2.2 ARITHMETIC & LOGIC UNIT (ALU)

The 8-bit arithmetic and logical instructions of the CPU are executed in the ALU. Internally the ALU communicates with the registers and the external data bus on the internal data bus. The type of functions performed by the ALU include:

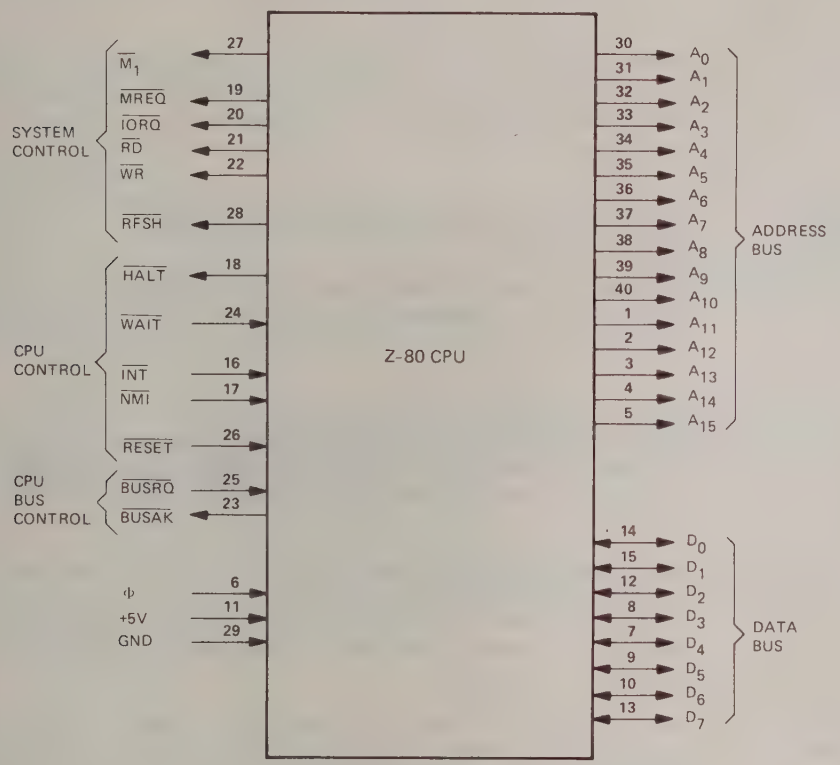
Add	Left or right shifts or rotates (arithmetic and logical)
Subtract	Increment
Logical AND	Decrement
Logical OR	Set bit
Logical Exclusive OR	Reset bit
Compare	Test bit

2.3 INSTRUCTION REGISTER AND CPU CONTROL

As each instruction is fetched from memory, it is placed in the instruction register and decoded. The control sections performs this function and then generates and supplies all of the control signals necessary to read or write data from or to the registers, control the ALU and provide all required external control signals.

3.0 Z-80 CPU PIN DESCRIPTION

The Z-80 CPU is packaged in an industry standard 40 pin Dual In-Line Package. The I/O pins are shown in figure 3.0-1 and the function of each is described below.



Z-80 PIN CONFIGURATION
FIGURE 3.0-1

A_0-A_{15}
(Address Bus)

Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanges and for I/O device data exchanges. I/O addressing uses the 8 lower address bits to allow the user to directly select up to 256 input or 256 output ports. A_0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address.

D_0-D_7
(Data Bus)

Tri-state input/output, active high. D_0-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$
(Machine Cycle one)

Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution. Note that during execution of 2-byte op-codes, $\overline{M1}$ is generated as each op code byte is fetched. These two byte op-codes always begin with CBH, DDH, EDH or FDH. $\overline{M1}$ also occurs with $\overline{IORQ}$ to indicate an interrupt acknowledge cycle.

$\overline{MREQ}$
(Memory Request)

Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{\text{IORQ}}$ (Input/Output Request)	Tri-state output, active low. The $\overline{\text{IORQ}}$ signal indicates that the lower half of the address bus holds a valid I/O address for a I/O read or write operation. An $\overline{\text{IORQ}}$ signal is also generated with an $\overline{\text{M1}}$ signal when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus. Interrupt Acknowledge operations occur during M_1 time while I/O operations never occur during M_1 time.
$\overline{\text{RD}}$ (Memory Read)	Tri-state output, active low. $\overline{\text{RD}}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.
$\overline{\text{WR}}$ (Memory Write)	Tri-state output, active low. $\overline{\text{WR}}$ indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.
$\overline{\text{RFSH}}$ (Refresh)	Output, active low. $\overline{\text{RFSH}}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current $\overline{\text{MREQ}}$ signal should be used to do a refresh read to all dynamic memories.
$\overline{\text{HALT}}$ (Halt state)	Output, active low. $\overline{\text{HALT}}$ indicates that the CPU has executed a HALT software instruction and is awaiting either a non maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOP's to maintain memory refresh activity.
$\overline{\text{WAIT}}$ (Wait)	Input, active low. $\overline{\text{WAIT}}$ indicates to the Z-80 CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active. This signal allows memory or I/O devices of any speed to be synchronized to the CPU.
$\overline{\text{INT}}$ (Interrupt Request)	Input, active low. The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled and if the $\overline{\text{BUSRQ}}$ signal is not active. When the CPU accepts the interrupt, an acknowledge signal ($\overline{\text{IORQ}}$ during M_1 time) is sent out at the beginning of the next instruction cycle. The CPU can respond to an interrupt in three different modes that are described in detail in section 5.4 (CPU Control Instructions).
$\overline{\text{NMI}}$ (Non Maskable Interrupt)	Input, negative edge triggered. The non maskable interrupt request line has a higher priority than $\overline{\text{INT}}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{\text{NMI}}$ automatically forces the Z-80 CPU to restart to location 0066_H. The program counter is automatically saved in the external stack so that the user can return to the program that was interrupted. Note that continuous WAIT cycles can prevent the current instruction from ending, and that a $\overline{\text{BUSRQ}}$ will override a $\overline{\text{NMI}}$.

$\overline{\text{RESET}}$

Input, active low. $\overline{\text{RESET}}$ forces the program counter to zero and initializes the CPU. The CPU initialization includes:

1) Disable the interrupt enable flip-flop
2) Set Register I = 00_H
3) Set Register R = 00_H
4) Set Interrupt Mode 0

During reset time, the address bus and data bus go to a high impedance state and all control output signals go to the inactive state.

$\overline{\text{BUSRQ}}$
(Bus Request)

Input, active low. The bus request signal is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these buses. When $\overline{\text{BUSRQ}}$ is activated, the CPU will set these buses to a high impedance state as soon as the current CPU machine cycle is terminated.

$\overline{\text{BUSAK}}$
(Bus Acknowledge)

Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.

Φ

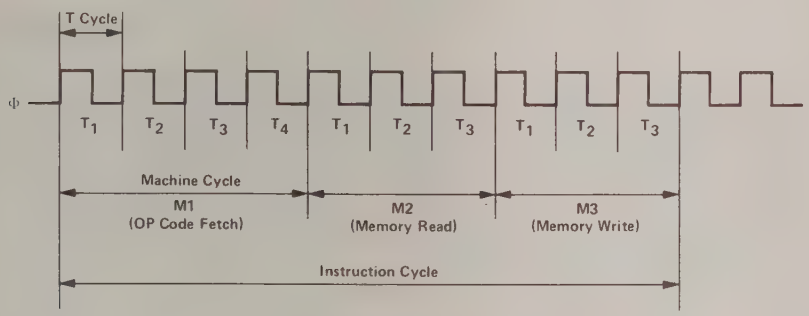
Single phase TTL level clock which requires only a 330 ohm pull-up resistor to +5 volts to meet all clock requirements.

4.0 CPU TIMING

The Z-80 CPU executes instructions by stepping through a very precise set of a few basic operations. These include:

Memory read or write
I/O device read or write
Interrupt acknowledge

All instructions are merely a series of these basic operations. Each of these basic operations can take from three to six clock periods to complete or they can be lengthened to synchronize the CPU to the speed of external devices. The basic clock periods are referred to as T cycles and the basic operations are referred to as M (for machine) cycles. Figure 4.0-0 illustrates how a typical instruction will be merely a series of specific M and T cycles. Notice that this instruction consists of three machine cycles (M1, M2 and M3). The first machine cycle of any instruction is a fetch cycle which is four, five or six T cycles long (unless lengthened by the wait signal which will be fully described in the next section). The fetch cycle (M1) is used to fetch the OP code of the next instruction to be executed. Subsequent machine cycles move data between the CPU and memory or I/O devices and they may have anywhere from three to five T cycles (again they may be lengthened by wait states to synchronize the external devices to the CPU). The following paragraphs describe the timing which occurs within any of the basic machine cycles. In section 10, the exact timing for each instruction is specified.



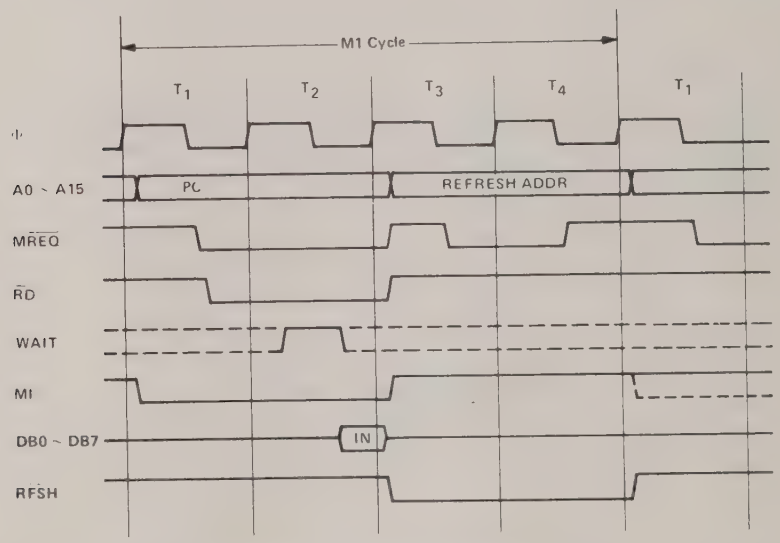
BASIC CPU TIMING EXAMPLE
FIGURE 4.0-0

All CPU timing can be broken down into a few very simple timing diagrams as shown in figure 4.0-1 through 4.0-7. These diagrams show the following basic operations with and without wait states (wait states are added to synchronize the CPU to slow memory or I/O devices).

4.0-1. Instruction OP code fetch (M1 cycle)
4.0-2. Memory data read or write cycles
4.0-3. I/O read or write cycles
4.0-4. Bus Request/Acknowledge Cycle
4.0-5. Interrupt Request/Acknowledge Cycle
4.0-6. Non maskable Interrupt Request/Acknowledge Cycle
4.0-7. Exit from a HALT instruction

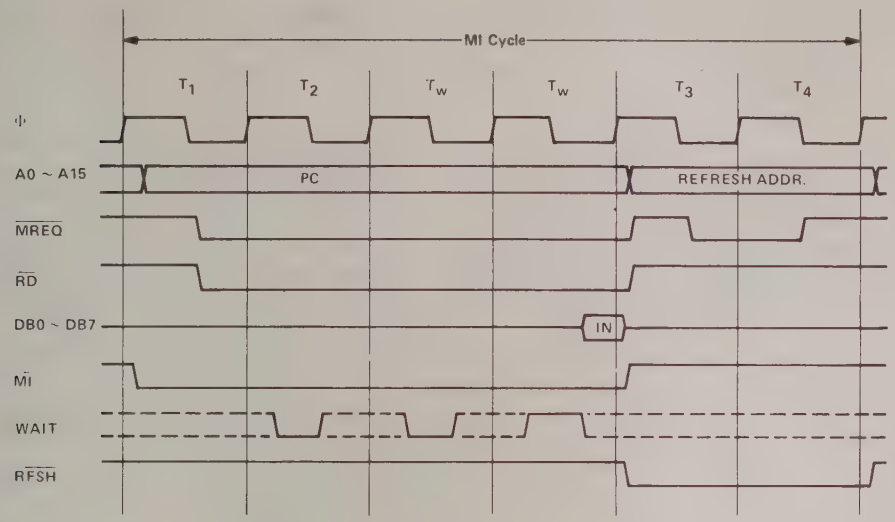
INSTRUCTION FETCH

Figure 4.0-1 shows the timing during an M1 cycle (OP code fetch). Notice that the PC is placed on the address bus at the beginning of the M1 cycle. One half clock time later the $\overline{MREQ}$ signal goes active. At this time the address to the memory has had time to stabilize so that the falling edge of $\overline{MREQ}$ can be used directly as a chip enable clock to dynamic memories. The $\overline{RD}$ line also goes active to indicate that the memory read data should be enabled onto the CPU data bus. The CPU samples the data from the memory on the data bus with the rising edge of the clock of state T3 and this same edge is used by the CPU to turn off the $\overline{RD}$ and $\overline{MRQ}$ signals. Thus the data has already been sampled by the CPU before the $\overline{RD}$ signal becomes inactive. Clock state T3 and T4 of a fetch cycle are used to refresh dynamic memories. (The CPU uses this time to decode and execute the fetched instruction so that no other operation could be performed at this time). During T3 and T4 the lower 7 bits of the address bus contain a memory refresh address and the $\overline{RFSH}$ signal becomes active to indicate that a refresh read of all dynamic memories should be accomplished. Notice that a $\overline{RD}$ signal is not generated during refresh time to prevent data from different memory segments from being gated onto the data bus. The $\overline{MREQ}$ signal during refresh time should be used to perform a refresh read of all memory elements. The refresh signal can not be used by itself since the refresh address is only guaranteed to be stable during $\overline{MREQ}$ time.



INSTRUCTION OP CODE FETCH
FIGURE 4.0-1

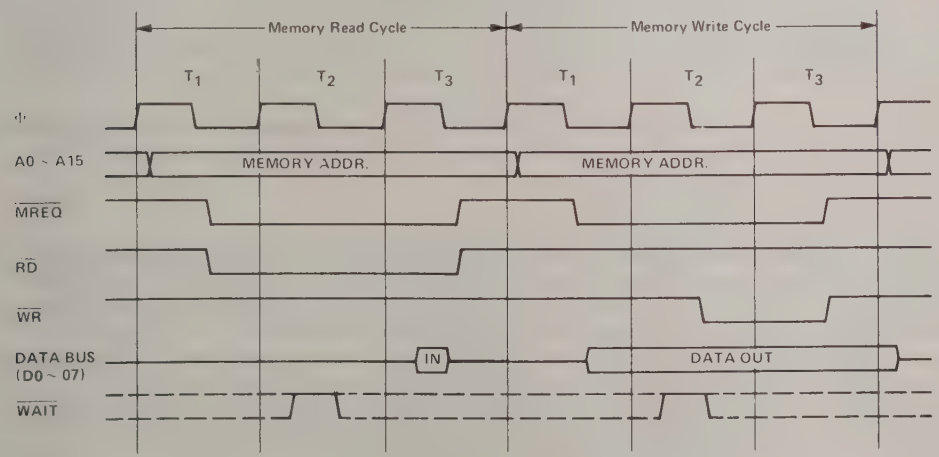
Figure 4.0-1A illustrates how the fetch cycle is delayed if the memory activates the $\overline{WAIT}$ line. During T2 and every subsequent Tw, the CPU samples the $\overline{WAIT}$ line with the falling edge of Φ. If the $\overline{WAIT}$ line is active at this time, another wait state will be entered during the following cycle. Using this technique the read cycle can be lengthened to match the access time of any type of memory device.



INSTRUCTION OP CODE FETCH WITH WAIT STATES
FIGURE 4.0-1A

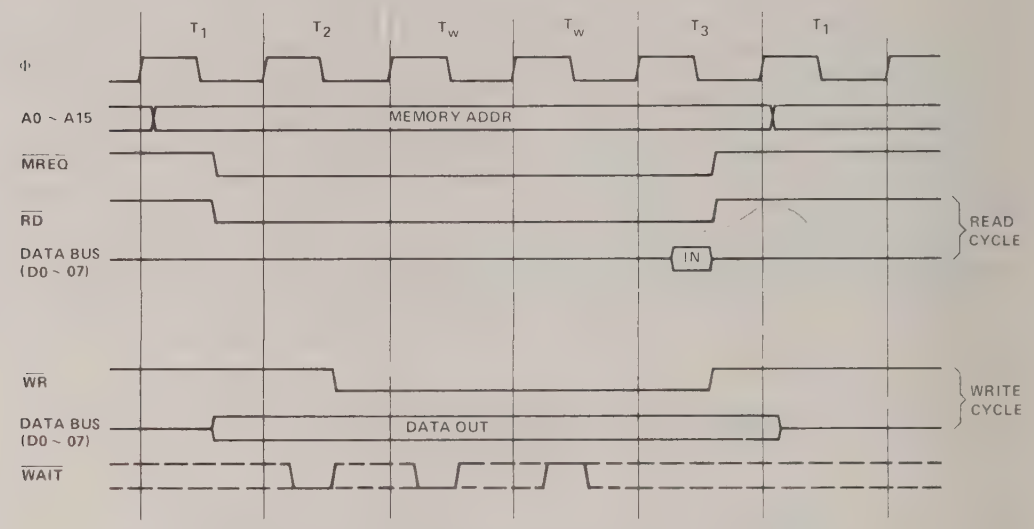
MEMORY READ OR WRITE

Figure 4.0-2 illustrates the timing of memory read or write cycles other than an OP code fetch (M1 cycle). These cycles are generally three clock periods long unless wait states are requested by the memory via the WAIT signal. The MREQ signal and the RD signal are used the same as in the fetch cycle. In the case of a memory write cycle, the MREQ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The WR line is active when data on the data bus is stable so that it can be used directly as a R/W pulse to virtually any type of semiconductor memory. Furthermore the WR signal goes inactive one half T state before the address and data bus contents are changed so that the overlap requirements for virtually any type of semiconductor memory type will be met.



MEMORY READ OR WRITE CYCLES
FIGURE 4.0-2

Figure 4.0-2A illustrates how a $\overline{\text{WAIT}}$ request signal will lengthen any memory read or write operation. This operation is identical to that previously described for a fetch cycle. Notice in this figure that a separate read and a separate write cycle are shown in the same figure although read and write cycles can never occur simultaneously.



MEMORY READ OR WRITE CYCLES WITH WAIT STATES
FIGURE 4.0-2A

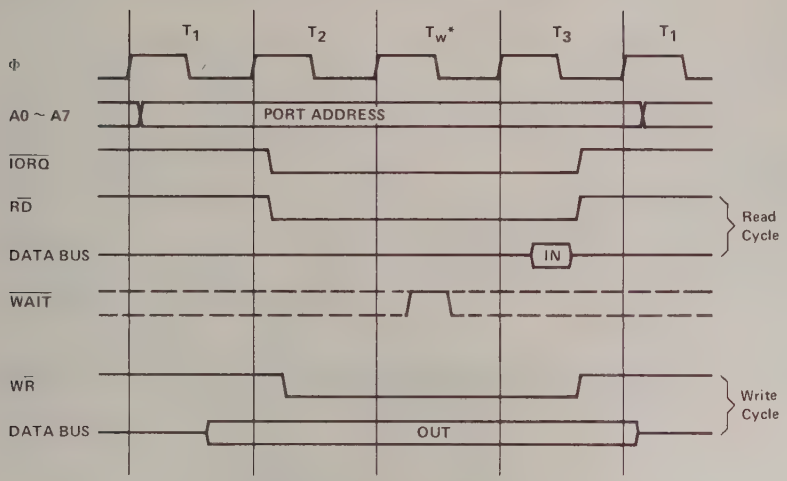
INPUT OR OUTPUT CYCLES

Figure 4.0-3 illustrates an I/O read or I/O write operation. Notice that during I/O operations a single wait state is automatically inserted. The reason for this is that during I/O operations, the time from when the $\overline{\text{IORQ}}$ signal goes active until the CPU must sample the $\overline{\text{WAIT}}$ line is very short and without this extra state sufficient time does not exist for an I/O port to decode its address and activate the $\overline{\text{WAIT}}$ line if a wait is required. Also, without this wait state it is difficult to design MOS I/O devices that can operate at full CPU speed. During this wait state time the $\overline{\text{WAIT}}$ request signal is sampled. During a read I/O operation, the $\overline{\text{RD}}$ line is used to enable the addressed port onto the data bus just as in the case of a memory read. For I/O write operations, the $\overline{\text{WR}}$ line is used as a clock to the I/O port, again with sufficient overlap timing automatically provided so that the rising edge may be used as a data clock.

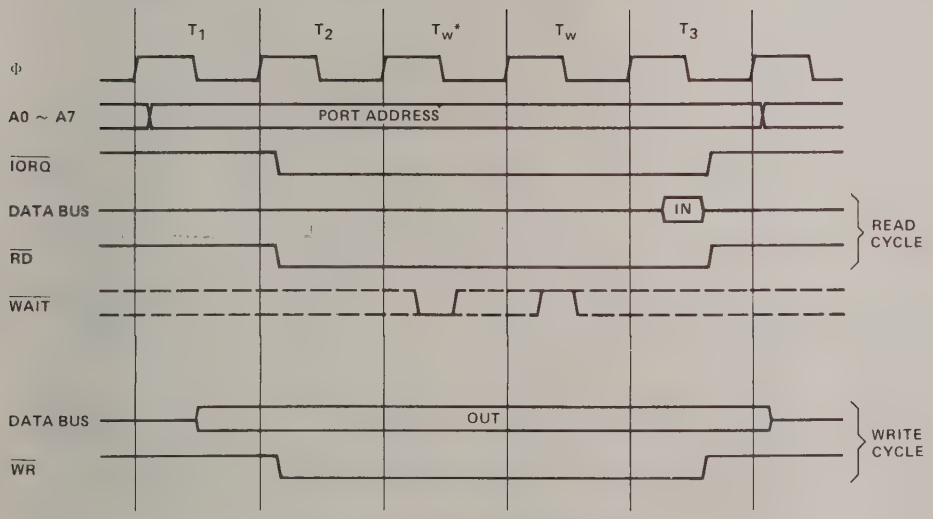
Figure 4.0-3A illustrates how additional wait states may be added with the $\overline{\text{WAIT}}$ line. The operation is identical to that previously described.

BUS REQUEST/ACKNOWLEDGE CYCLE

Figure 4.0-4 illustrates the timing for a Bus Request/Acknowledge cycle. The $\overline{\text{BUSRQ}}$ signal is sampled by the CPU with the rising edge of the last clock period of any machine cycle. If the $\overline{\text{BUSRQ}}$ signal is active, the CPU will set its address, data and tri-state control signals to the high impedance state with the rising edge of the next clock pulse. At that time any external device can control the buses to transfer data between memory and I/O devices. (This is generally known as Direct Memory Access [DMA] using cycle stealing). The maximum time for the CPU to respond to a bus request is the length of a machine cycle and the external controller can maintain control of the bus for as many clock cycles as is desired. Note, however, that if very long DMA cycles are used, and dynamic memories are being used, the external controller must also perform the refresh function. This situation only occurs if very large blocks of data are transferred under DMA control. Also note that during a bus request cycle, the CPU cannot be interrupted by either a $\overline{\text{NMI}}$ or an $\overline{\text{INT}}$ signal.

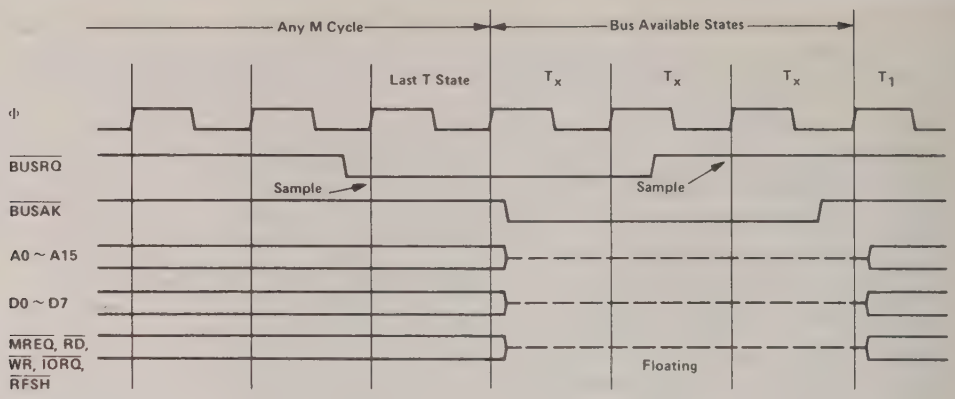


INPUT OR OUTPUT CYCLES
FIGURE 4.0-3



INPUT OR OUTPUT CYCLES WITH WAIT STATES
FIGURE 4.0-3A

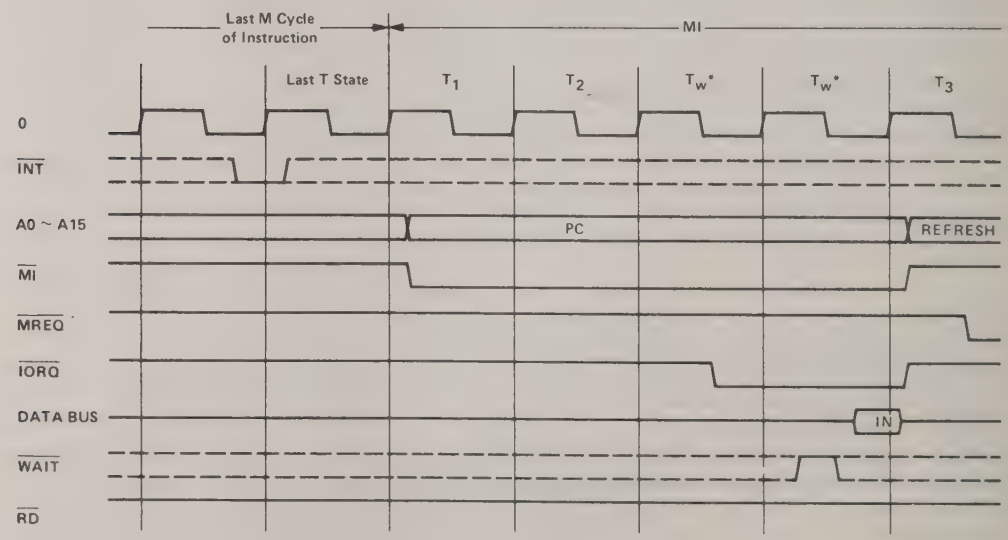
* Automatically inserted WAIT state



BUS REQUEST/ACKNOWLEDGE CYCLE
FIGURE 4.0-4

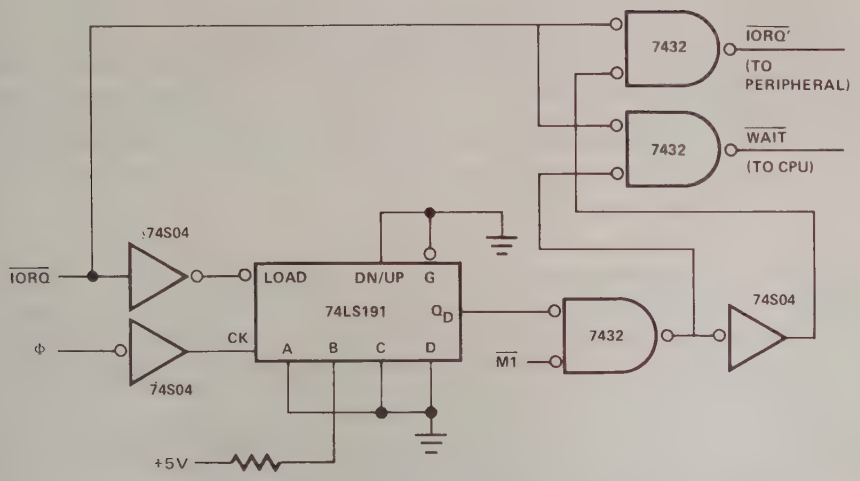
INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

Figure 4.0-5 illustrates the timing associated with an interrupt cycle. The interrupt signal ($\overline{INT}$) is sampled by the CPU with the rising edge of the last clock at the end of any instruction. The signal will not be accepted if the internal CPU software controlled interrupt enable flip-flop is not set or if the $\overline{BUSRQ}$ signal is active. When the signal is accepted a special M1 cycle is generated. During this special M1 cycle the $\overline{IORQ}$ signal becomes active (instead of the normal $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. Notice that two wait states are automatically added to this cycle. These states are added so that a ripple priority interrupt scheme can be easily implemented. The two wait states allow sufficient time for the ripple signals to stabilize and identify which I/O device must insert the response vector. Refer to section 8.0 for details on how the interrupt response vector is utilized by the CPU.

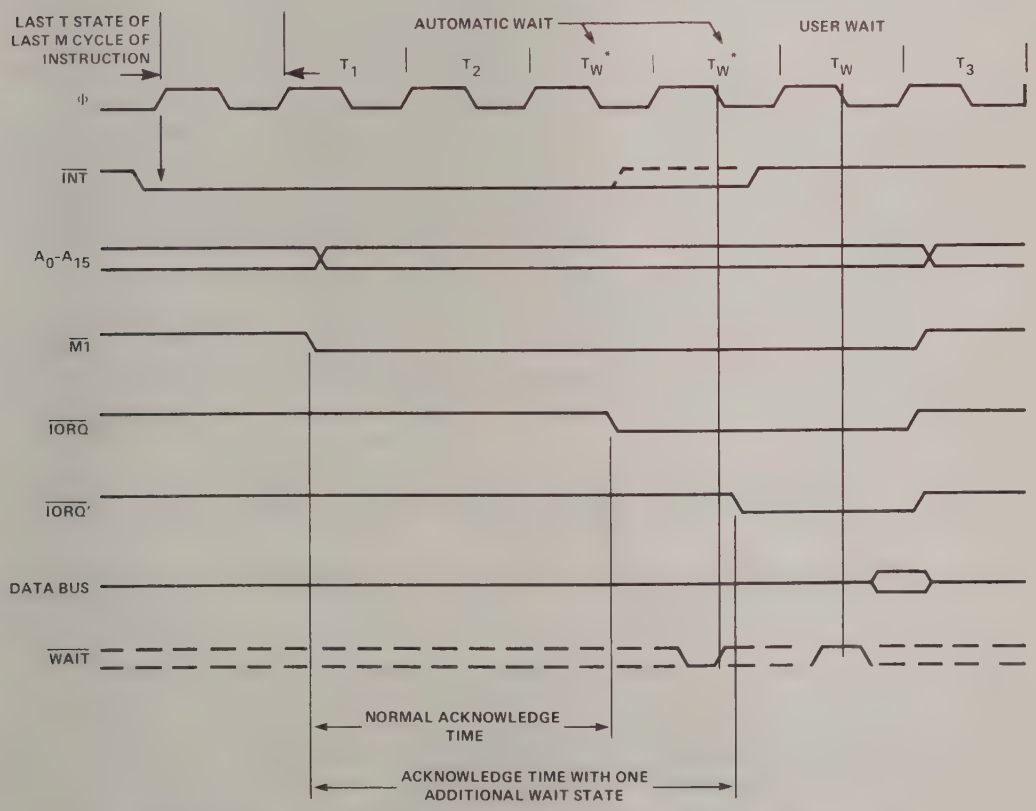


INTERRUPT REQUEST/ACKNOWLEDGE CYCLE
FIGURE 4.0-5

Figures 4.0-5A and 4.0-5B illustrate how a programmable counter can be used to extend interrupt acknowledge time. (Configured as shown to add one wait state)



EXTENDING INTERRUPT ACKNOWLEDGE TIME WITH WAIT STATE
FIGURE 4.0-5A



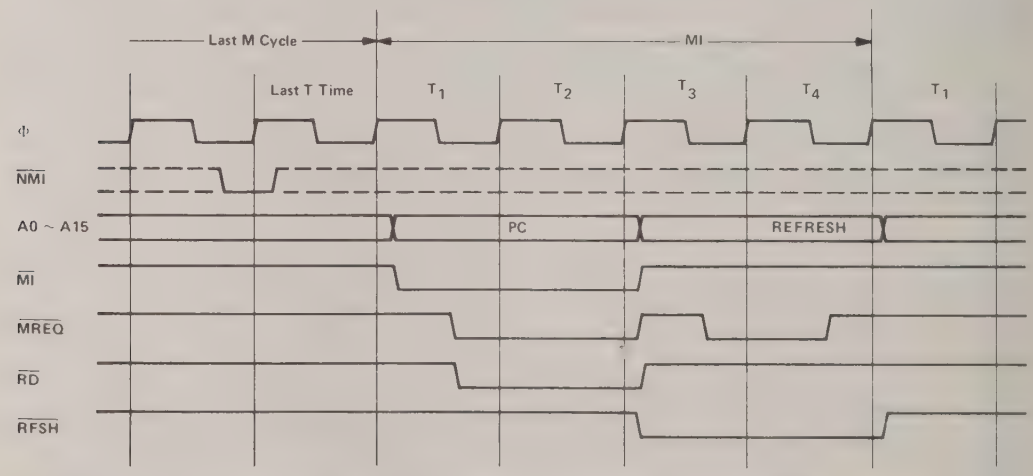
REQUEST/ACKNOWLEDGE CYCLE WITH ONE ADDITIONAL WAIT STATE
FIGURE 4.0-5B

NON MASKABLE INTERRUPT RESPONSE

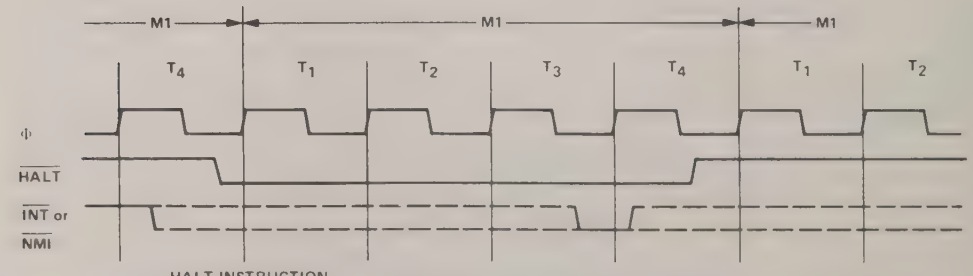
Figure 4.0-6 illustrates the request/acknowledge cycle for the non maskable interrupt. This signal is sampled at the same time as the interrupt line, but this line has priority over the normal interrupt and it can not be disabled under software control. Its usual function is to provide immediate response to important signals such as an impending power failure. The CPU response to a non maskable interrupt is similar to a normal memory read operation. The only difference being that the content of the data bus is ignored while the processor automatically stores the PC in the external stack and jumps to location 0066_H. The service routine for the non maskable interrupt must begin at this location if this interrupt is used.

HALT EXIT

Whenever a software halt instruction is executed the CPU begins executing NOP's until an interrupt is received (either a non maskable or a maskable interrupt while the interrupt flip flop is enabled). The two interrupt lines are sampled with the rising clock edge during each T4 state as shown in figure 4.0-7. If a non maskable interrupt has been received or a maskable interrupt has been received and the interrupt enable flip-flop is set, then the halt state will be exited on the next rising clock edge. The following cycle will then be an interrupt acknowledge cycle corresponding to the type of interrupt that was received. If both are received at this time, then the non maskable one will be acknowledged since it has highest priority. The purpose of executing NOP instructions while in the halt state is to keep the memory refresh signals active. Each cycle in the halt state is a normal M1 (fetch) cycle except that the data received from the memory is ignored and a NOP instruction is forced internally to the CPU. The halt acknowledge signal is active during this time to indicate that the processor is in the halt state.



NON MASKABLE INTERRUPT REQUEST OPERATION
FIGURE 4.0-6



HALT EXIT
FIGURE 4.0-7

5.0 Z-80 CPU INSTRUCTION SET

The Z-80 CPU can execute 158 different instruction types including all 78 of the 8080A CPU. The instructions can be broken down into the following major groups:

- Load and Exchange
- Block Transfer and Search
- Arithmetic and Logical
- Rotate and Shift
- Bit Manipulation (set, reset, test)
- Jump, Call and Return
- Input/Output
- Basic CPU Control

5.1 INTRODUCTION TO INSTRUCTION TYPES

The load instructions move data internally between CPU registers or between CPU registers and external memory. All of these instructions must specify a source location from which the data is to be moved and a destination location. The source location is not altered by a load instruction. Examples of load group instructions include moves between any of the general purpose registers such as move the data to Register B from Register C. This group also includes load immediate to any CPU register or to any external memory location. Other types of load instructions allow transfer between CPU registers and memory locations. The exchange instructions can trade the contents of two registers.

A unique set of block transfer instructions is provided in the Z-80. With a single instruction a block of memory of any size can be moved to any other location in memory. This set of block moves is extremely valuable when large strings of data must be processed. The Z-80 block search instructions are also valuable for this type of processing. With a single instruction, a block of external memory of any desired length can be searched for any 8-bit character. Once the character is found or the end of the block is reached, the instruction automatically terminates. Both the block transfer and the block search instructions can be interrupted during their execution so as to not occupy the CPU for long periods of time.

The arithmetic and logical instructions operate on data stored in the accumulator and other general purpose CPU registers or external memory locations. The results of the operations are placed in the accumulator and the appropriate flags are set according to the result of the operation. An example of an arithmetic operation is adding the accumulator to the contents of an external memory location. The results of the addition are placed in the accumulator. This group also includes 16-bit addition and subtraction between 16-bit CPU registers.

The rotate and shift group allows any register or any memory location to be rotated right or left with or without carry either arithmetic or logical. Also, a digit in the accumulator can be rotated right or left with two digits in any memory location.

The bit manipulation instructions allow any bit in the accumulator, any general purpose register or any external memory location to be set, reset or tested with a single instruction. For example, the most significant bit of register H can be reset. This group is especially useful in control applications and for controlling software flags in general purpose programming.

The jump, call and return instructions are used to transfer between various locations in the user's program. This group uses several different techniques for obtaining the new program counter address from specific external memory locations. A unique type of call is the restart instruction. This instruction actually contains the new address as a part of the 8-bit OP code. This is possible since only 8 separate addresses located in page zero of the external memory may be specified. Program jumps may also be achieved by loading register HL, IX or IY directly into the PC, thus allowing the jump address to be a complex function of the routine being executed.

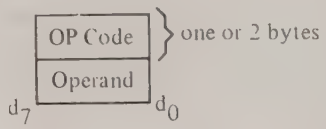
The input/output group of instructions in the Z-80 allow for a wide range of transfers between external memory locations or the general purpose CPU registers, and the external I/O devices. In each case, the port number is provided on the lower 8 bits of the address bus during any I/O transaction. One instruction allows this port number to be specified by the second byte of the instruction while other Z-80 instructions allow it to be specified as the content of the C register. One major advantage of using the C register as a pointer to the I/O device is that it allows different I/O ports to share common software driver routines. This is not possible when the address is part of the OP code if the routines are stored in ROM. Another feature of these input instructions is that they set the flag register automatically so that additional operations are not required to determine the state of the input data (for example its parity). The Z-80 CPU includes single instructions that can move blocks of data (up to 256 bytes) automatically to or from any I/O port directly to any memory location. In conjunction with the dual set of general purpose registers, these instructions provide for fast I/O block transfer rates. The value of this I/O instruction set is demonstrated by the fact that the Z-80 CPU can provide all required floppy disk formatting (i.e., the CPU provides the preamble, address, data and enables the CRC codes) on double density floppy disk drives on an interrupt driven basis.

Finally, the basic CPU control instructions allow various options and modes. This group includes instructions such as setting or resetting the interrupt enable flip flop or setting the mode of interrupt response.

5.2 ADDRESSING MODES

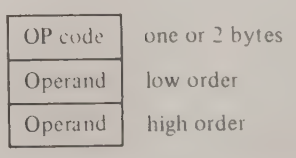
Most of the Z-80 instructions operate on data stored in internal CPU registers, external memory or in the I/O ports. Addressing refers to how the address of this data is generated in each instruction. This section gives a brief summary of the types of addressing used in the Z-80 while subsequent sections detail the type of addressing available for each instruction group.

Immediate. In this mode of addressing the byte following the OP code in memory contains the actual operand.



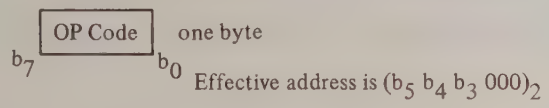
Examples of this type of instruction would be to load the accumulator with a constant, where the constant is the byte immediately following the OP code.

Immediate Extended. This mode is merely an extension of immediate addressing in that the two bytes following the OP codes are the operand.

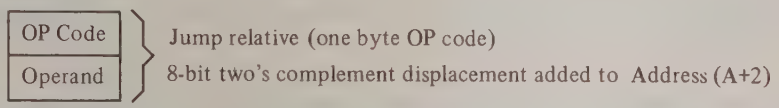


Examples of this type of instruction would be to load the HL register pair (16-bit register) with 16 bits (2 bytes) of data.

Modified Page Zero Addressing. The Z-80 has a special single byte CALL instruction to any of 8 locations in page zero of memory. This instruction (which is referred to as a restart) sets the PC to an effective address in page zero. The value of this instruction is that it allows a single byte to specify a complete 16-bit address where commonly called subroutines are located, thus saving memory space.

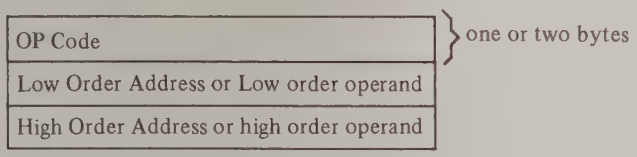


Relative Addressing. Relative addressing uses one byte of data following the OP code to specify a displacement from the existing program to which a program jump can occur. This displacement is a signed two's complement number that is added to the address of the OP code of the following instruction.



The value of relative addressing is that it allows jumps to nearby locations while only requiring two bytes of memory space. For most programs, relative jumps are by far the most prevalent type of jump due to the proximity of related program segments. Thus, these instructions can significantly reduce memory space requirements. The signed displacement can range between +127 and -128 from A + 2. This allows for a total displacement of +129 to -126 from the jump relative OP code address. Another major advantage is that it allows for relocatable code.

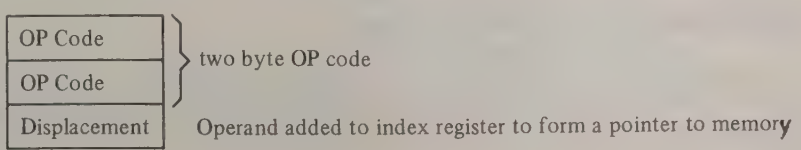
Extended Addressing. Extended Addressing provides for two bytes (16 bits) of address to be included in the instruction. This data can be an address to which a program can jump or it can be an address where an operand is located.



Extended addressing is required for a program to jump from any location in memory to any other location, or load and store data in any memory location.

When extended addressing is used to specify the source or destination address of an operand, the notation (nn) will be used to indicate the content of memory at nn, where nn is the 16-bit address specified in the instruction. This means that the two bytes of address nn are used as a pointer to a memory location. The use of the parentheses always means that the value enclosed within them is used as a pointer to a memory location. For example, (1200) refers to the contents of memory at location 1200.

Indexed Addressing. In this type of addressing, the byte of data following the OP code contains a displacement which is added to one of the two index registers (the OP code specifies which index register is used) to form a pointer to memory. The contents of the index register are not altered by this operation.



An example of an indexed instruction would be to load the contents of the memory location (Index Register + Displacement) into the accumulator. The displacement is a signed two's complement number. Indexed addressing greatly simplifies programs using tables of data since the index register can point to the start of any table. Two index registers are provided since very often operations require two or more tables. Indexed addressing also allows for relocatable code.

The two index registers in the Z-80 are referred to as IX and IY. To indicate indexed addressing the notation:

(IX+d) or (IY+d)

is used. Here d is the displacement specified after the OP code. The parentheses indicate that this value is used as a pointer to external memory.

Register Addressing. Many of the Z-80 OP codes contain bits of information that specify which CPU register is to be used for an operation. An example of register addressing would be to load the data in register B into register C.

Implied Addressing. Implied addressing refers to operations where the OP code automatically implies one or more CPU registers as containing the operands. An example is the set of arithmetic operations where the accumulator is always implied to be the destination of the results.

Register Indirect Addressing. This type of addressing specifies a 16-bit CPU register pair (such as HL) to be used as a pointer to any location in memory. This type of instruction is very powerful and it is used in a wide range of applications.

An example of this type of instruction would be to load the accumulator with the data in the memory location pointed to by the HL register contents. Indexed addressing is actually a form of register indirect addressing except that a displacement is added with indexed addressing. Register indirect addressing allows for very powerful but simple to implement memory accesses. The block move and search commands in the Z-80 are extensions of this type of addressing where automatic register incrementing, decrementing and comparing has been added. The notation for indicating register indirect addressing is to put parentheses around the name of the register that is to be used as the pointer. For example, the symbol

(HL)

specifies that the contents of the HL register are to be used as a pointer to a memory location. Often register indirect addressing is used to specify 16-bit operands. In this case, the register contents point to the lower order portion of the operand while the register contents are automatically incremented to obtain the upper portion of the operand.

Bit Addressing. The Z-80 contains a large number of bit set, reset and test instructions. These instructions allow any memory location or CPU register to be specified for a bit operation through one of three previous addressing modes (register, register indirect and indexed) while three bits in the OP code specify which of the eight bits is to be manipulated.

ADDRESSING MODE COMBINATIONS

Many instructions include more than one operand (such as arithmetic instructions or loads). In these cases, two types of addressing may be employed. For example, load can use immediate addressing to specify the source and register indirect or indexed addressing to specify the destination.

5.3 INSTRUCTION OP CODES

This section describes each of the Z-80 instructions and provides tables listing the OP codes for every instruction. In each of these tables the OP codes in bold type are identical to those offered in the 8080A CPU. Also shown is the assembly language mnemonic that is used for each instruction. All instruction OP codes are listed in hexadecimal notation. Single byte OP codes require two hex characters while double byte OP codes require four hex characters. The conversion from hex to binary is repeated here for convenience.

Hex		Binary		Decimal	Hex		Binary		Decimal
0	=	0000	=	0	8	=	1000	=	8
1	=	0001	=	1	9	=	1001	=	9
2	=	0010	=	2	A	=	1010	=	10
3	=	0011	=	3	B	=	1011	=	11
4	=	0100	=	4	C	=	1100	=	12
5	=	0101	=	5	D	=	1101	=	13
6	=	0110	=	6	E	=	1110	=	14
7	=	0111	=	7	F	=	1111	=	15

Z-80 instruction mnemonics consist of an OP code and zero, one or two operands. Instructions in which the operand is implied have no operand. Instructions which have only one logical operand or those in which one operand is invariant (such as the Logical OR instruction) are represented by a one operand mnemonic. Instructions which may have two varying operands are represented by two operand mnemonics.

LOAD AND EXCHANGE

Table 5.3-1 defines the OP code for all of the 8-bit load instructions implemented in the Z-80 CPU. Also shown in this table is the type of addressing used for each instruction. The source of the data is found on the top horizontal row while the destination is specified by the left hand column. For example, load register C from register B uses the OP code 48H. In all of the tables the OP code is specified in hexadecimal notation and the 48H (=0100 1000 binary) code is fetched by the CPU from the external memory during M1 time, decoded and then the register transfer is automatically performed by the CPU.

The assembly language mnemonic for this entire group is LD, followed by the destination followed by the source (LD DEST., SOURCE). Note that several combinations of addressing modes are possible. For example, the source may use register addressing and the destination may be register indirect; such as load the memory location pointed to by register HL with the contents of register D. The OP code for this operation would be 72. The mnemonic for this load instruction would be as follows:

LD (HL), D

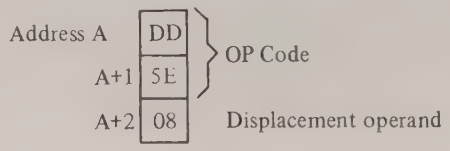
The parentheses around the HL means that the contents of HL are used as a pointer to a memory location. In all Z-80 load instruction mnemonics the destination is always listed first, with the source following. The Z-80 assembly language has been defined for ease of programming. Every instruction is self documenting and programs written in Z-80 language are easy to maintain.

Note in table 5.3-1 that some load OP codes that are available in the Z-80 use two bytes. This is an efficient method of memory utilization since 8, 16, 24 or 32 bit instructions are implemented in the Z-80. Thus often utilized instructions such as arithmetic or logical operations are only 8-bits which results in better memory utilization than is achieved with fixed instruction sizes such as 16-bits.

All load instructions using indexed addressing for either the source or destination location actually use three bytes of memory with the third byte being the displacement d. For example a load register E with the operand pointed to by IX with an offset of +8 would be written:

LD E, (IX + 8)

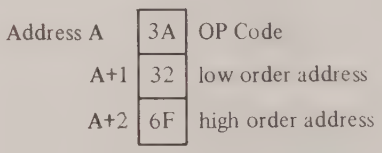
The instruction sequence for this in memory would be:



The two extended addressing instructions are also three byte instructions. For example the instruction to load the accumulator with the operand in memory location 6F32H would be written:

LD A, (6F 32H)

and its instruction sequence would be:

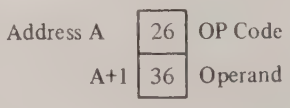


Notice that the low order portion of the address is always the first operand.

The load immediate instructions for the general purpose 8-bit registers are two-byte instructions. The instruction load register H with the value 36H would be written:

LD H, 36H

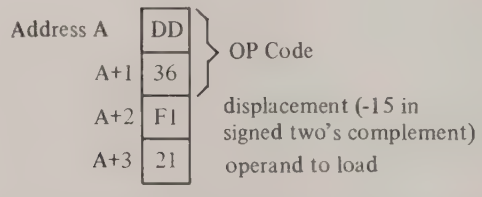
and its sequence would be:



Loading a memory location using indexed addressing for the destination and immediate addressing for the source requires four bytes. For example:

LD (IX - 15), 21H

would appear as:



Notice that with any indexed addressing the displacement always follows directly after the OP code.

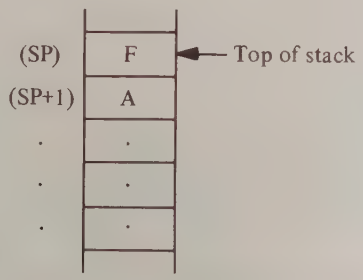
Table 5.3-2 specifies the 16-bit load operations. This table is very similar to the previous one. Notice that the extended addressing capability covers all register pairs. Also notice that register indirect operations specifying the stack pointer are the PUSH and POP instructions. The mnemonic for these instructions is "PUSH" and "POP." These differ from other 16-bit loads in that the stack pointer is automatically decremented and incremented as each byte is pushed onto or popped from the stack respectively. For example the instruction:

PUSH AF

is a single byte instruction with the OP code of F5H. When this instruction is executed the following sequence is generated:

Decrement SP
LD (SP), A
Decrement SP
LD (SP), F

Thus the external stack now appears as follows:



			SOURCE															
			IMPLIED		REGISTER							REG INDIRECT			INDEXED		EXT. ADDR.	IMME.
			I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX + d)	(IY + d)	(nn)	n
DESTINATION	REGISTER	A	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
		B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
		C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
		D			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
		E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
		H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
		L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
	REG INDIRECT	(HL)			77	70	71	72	73	74	75							36 n
		(BC)			02													
		(DE)			12													
	INDEXED	(IX+d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
		(IY+d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
	EXT. ADDR	(nn)			32 n n													
	IMPLIED	I			ED 47													
		R			ED 4F													

8 BIT LOAD GROUP
'LD'
TABLE 5.3–1

The POP instruction is the exact reverse of a PUSH. Notice that all PUSH and POP instructions utilize a 16-bit operand and the high order byte is always pushed first and popped last. That is a:

PUSH BC is PUSH B then C
PUSH DE is PUSH D then E
PUSH HL is PUSH H then L
POP HL is POP L then H

The instruction using extended immediate addressing for the source obviously requires 2 bytes of data following the OP code. For example:

LD DE, 0659H

will be:

Address A	11	OP Code
A+1	59	Low order operand to register E
A+2	06	High order operand to register D

In all extended immediate or extended addressing modes, the low order byte always appears first after the OP code.

Table 5.3-3 lists the 16-bit exchange instructions implemented in the Z-80. OP code 08H allows the programmer to switch between the two pairs of accumulator flag registers while D9H allows the programmer to switch between the duplicate set of six general purpose registers. These OP codes are only one byte in length to absolutely minimize the time necessary to perform the exchange so that the duplicate banks can be used to effect very fast interrupt response times.

BLOCK TRANSFER AND SEARCH

Table 5.3-4 lists the extremely powerful block transfer instructions. All of these instructions operate with three registers.

HL points to the source location.
DE points to the destination location.
BC is a byte counter.

After the programmer has initialized these three registers, any of these four instructions may be used. The LDI (Load and Increment) instruction moves one byte from the location pointed to by HL to the location pointed to by DE. Register pairs HL and DE are then automatically incremented and are ready to point to the following locations. The byte counter (register pair BC) is also decremented at this time. This instruction is valuable when blocks of data must be moved but other types of processing are required between each move. The LDIR (Load, increment and repeat) instruction is an extension of the LDI instruction. The same load and increment operation is repeated until the byte counter reaches the count of zero. Thus, this single instruction can move any block of data from one location to any other.

Note that since 16-bit registers are used, the size of the block can be up to 64K bytes (1K = 1024) long and it can be moved from any location in memory to any other location. Furthermore the blocks can be overlapping since there are absolutely no constraints on the data that is used in the three register pairs.

The LDD and LDDR instructions are very similar to the LDI and LDIR. The only difference is that register pairs HL and DE are decremented after every move so that a block transfer starts from the highest address of the designated block rather than the lowest.

DESTINATION		SOURCE: REGISTER AF	BC	DE	HL	SP	IX	IY	IMM. EXT. nn	EXT. ADDR. (nn)	REG. INDIR. (SP)
REGISTER	AF										**F1**
	BC								**01 n n**	ED 4B n n	**C1**
	DE								**11 n n**	ED 5B n n	**D1**
	HL								**21 n n**	**2A n n**	**E1**
	SP				**F9**		DD F9	FD F9	**31 n n**	ED 7B n n	
	IX								DD 21 n n	DD 2A n n	DD E1
	IY								FD 21 n n	FD 2A n n	FD E1
EXT. ADDR.	(nn)		ED 43 n n	ED 53 n n	**22 n n**	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH INSTRUCTIONS → REG. IND.	(SP)	**F5**	**C5**	**D5**	**E5**		DD E5	FD E5			

NOTE: The Push & Pop Instructions adjust the SP after every execution

POP INSTRUCTIONS

16 BIT LOAD GROUP
'LD'
'PUSH' AND 'POP'
TABLE 5.3–2

		IMPLIED ADDRESSING: AF′	BC′, DE′ & HL′	HL	IX	IY
IMPLIED	AF	08				
	BC, DE & HL		D9			
	DE			**EB**		
REG. INDIR.	(SP)			**E3**	DD E3	FD E3

EXCHANGES
'EX' AND 'EXX'
TABLE 5.3–3

		SOURCE		
		REG. INDIR.	(HL)	
DESTINATION	REG. INDIR.	(DE)	ED A0	'LDI' – Load (DE)←(HL) Inc HL & DE, Dec BC
			ED B0	'LDIR,' – Load (DE)←(HL) Inc HL & DE, Dec BC, Repeat until BC = 0
			ED A8	'LDD' – Load (DE)←(HL) Dec HL & DE, Dec BC
			ED B8	'LDDR' – Load (DE)←(HL) Dec HL & DE, Dec BC, Repeat until BC = 0

Reg HL points to source
Reg DE points to destination
Reg BC is byte counter

BLOCK TRANSFER GROUP
TABLE 5.3–4

Table 5.3-5 specifies the OP codes for the four block search instructions. The first, CPI (compare and increment) compares the data in the accumulator, with the contents of the memory location pointed to by register HL. The result of the compare is stored in one of the flag bits (see section 6.0 for a detailed explanation of the flag operations) and the HL register pair is then incremented and the byte counter (register pair BC) is decremented.

The instruction CPIR is merely an extension of the CPI instruction in which the compare is repeated until either a match is found or the byte counter (register pair BC) becomes zero. Thus, this single instruction can search the entire memory for any 8-bit character.

The CPD (Compare and Decrement) and CPDR (Compare, Decrement and Repeat) are similar instructions, their only difference being that they decrement HL after every compare so that they search the memory in the opposite direction. (The search is started at the highest location in the memory block).

It should be emphasized again that these block transfer and compare instructions are extremely powerful in string manipulation applications.

ARITHMETIC AND LOGICAL

Table 5.3-6 lists all of the 8-bit arithmetic operations that can be performed with the accumulator, also listed are the increment (INC) and decrement (DEC) instructions. In all of these instructions, except INC and DEC, the specified 8-bit operation is performed between the data in the accumulator and the source data specified in the table. The result of the operation is placed in the accumulator with the exception of compare (CP) that leaves the accumulator unaffected. All of these operations affect the flag register as a result of the specified operation. (Section 6.0 provides all of the details on how the flags are affected by any instruction type). INC and DEC instructions specify a register or a memory location as both source and destination of the result. When the source operand is addressed using the index registers the displacement must follow directly. With immediate addressing the actual operand will follow directly. For example the instruction:

AND 07H

would appear as:

Address A	E6	OP Code
A+1	07	Operand

SEARCH LOCATION REG. INDIR. (HL)	
ED A1	'CPI' Inc HL, Dec BC
ED B1	'CPIR', Inc HL, Dec BC repeat until BC = 0 or find match
ED A9	'CPD' Dec HL & BC
ED B9	'CPDR' Dec HL & BC Repeat until BC = 0 or find match

HL points to location in memory to be compared with accumulator contents
BC is byte counter

BLOCK SEARCH GROUP
TABLE 5.3–5

Assuming that the accumulator contained the value F3H the result of 03H would be placed in the accumulator:

Acc before operation	1111 0011 = F3H
Operand	0000 0111 = 07H
Result to Acc	0000 0011 = 03H

The Add instruction (ADD) performs a binary add between the data in the source location and the data in the accumulator. The subtract (SUB) does a binary subtraction. When the add with carry is specified (ADC) or the subtract with carry (SBC), then the carry flag is also added or subtracted respectively. The flags and decimal adjust instruction (DAA) in the Z-80 (fully described in section 6.0) allow arithmetic operations for:

multiprecision packed BCD numbers

multiprecision signed or unsigned binary numbers

multiprecision two's complement signed numbers

Other instructions in this group are logical and (AND), logical or (OR), exclusive or (XOR) and compare (CP).

There are five general purpose arithmetic instructions that operate on the accumulator or carry flag. These five are listed in table 5.3-7. The decimal adjust instruction can adjust for subtraction as well as addition, thus making BCD arithmetic operations simple. Note that to allow for this operation the flag N is used. This flag is set if the last arithmetic operation was a subtract. The negate accumulator (NEG) instruction forms the two's complement of the number in the accumulator. Finally notice that a reset carry instruction is not included in the Z-80 since this operation can be easily achieved through other instructions such as a logical AND of the accumulator with itself.

Table 5.3-8 lists all of the 16-bit arithmetic operations between 16-bit registers. There are five groups of instructions including add with carry and subtract with carry. ADC and SBC affect all of the flags. These two groups simplify address calculation operations or other 16-bit arithmetic operations.

SOURCE

	REGISTER ADDRESSING							REG. INDIR.	INDEXED		IMMED.
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADD w CARRY 'ADC'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARRY 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'AND'	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'XOR'	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'OR'	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

8 BIT ARITHMETIC AND LOGIC
TABLE 5.3–6

Decimal Adjust Acc, 'DAA'	27
Complement Acc, 'CPL'	2F
Negate Acc, 'NEG' (2's complement)	ED 44
Complement Carry Flag, 'CCF'	3F
Set Carry Flag, 'SCF'	37

GENERAL PURPOSE AF OPERATIONS
TABLE 5.3-7

DESTINATION		SOURCE: BC	DE	HL	SP	IX	IY
'ADD'	HL	09	19	29	39		
'ADD'	IX	DD 09	DD 19		DD 39	DD 29	
'ADD'	IY	FD 09	FD 19		FD 39		FD 29
ADD WITH CARRY AND SET FLAGS 'ADC'	HL	ED 4A	ED 5A	ED 6A	ED 7A		
SUB WITH CARRY AND SET FLAGS 'SBC'	HL	ED 42	ED 52	ED 62	ED 72		
INCREMENT 'INC'		03	13	23	33	DD 23	FD 23
DECREMENT 'DEC'		0B	1B	2B	3B	DD 2B	FD 2B

16 BIT ARITHMETIC
TABLE 5.3–8

ROTATE AND SHIFT

A major capability of the Z-80 is its ability to rotate or shift data in the accumulator, any general purpose register, or any memory location. All of the rotate and shift OP codes are shown in table 5.3-9. Also included in the Z-80 are arithmetic and logical shift operations. These operations are useful in an extremely wide range of applications including integer multiplication and division. Two BCD digit rotate instructions (RRD and RLD) allow a digit in the accumulator to be rotated with the two digits in a memory location pointed to by register pair HL. (See figure 5.3-9). These instructions allow for efficient BCD arithmetic.

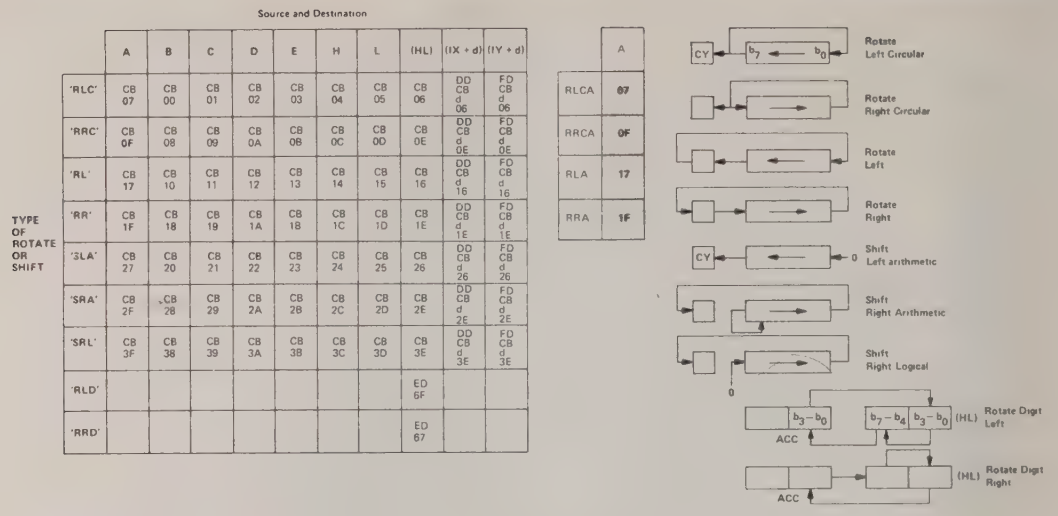
BIT MANIPULATION

The ability to set, reset and test individual bits in a register or memory location is needed in almost every program. These bits may be flags in a general purpose software routine, indications of external control conditions or data packed into memory locations to make memory utilization more efficient.

The Z-80 has the ability to set, reset or test any bit in the accumulator, any general purpose register or any memory location with a single instruction. Table 5.3-10 lists the 240 instructions that are available for this purpose. Register addressing can specify the accumulator or any general purpose register on which the operation is to be performed. Register indirect and indexed addressing are available to operate on external memory locations. Bit test operations set the zero flag (Z) if the tested bit is a zero. (Refer to section 6.0 for further explanation of flag operation).

JUMP, CALL AND RETURN

Figure 5.3-11 lists all of the jump, call and return instructions implemented in the Z-80 CPU. A jump is a branch in a program where the program counter is loaded with the 16-bit value as specified by one of the three available addressing modes (Immediate Extended, Relative or Register Indirect). Notice that the jump group has several different conditions that can be specified to be met before the jump will be made. If these conditions are not met, the program merely continues with the next sequential instruction. The conditions are all dependent on the data in the flag register. (Refer to section 6.0 for details on the flag register). The immediate extended addressing is used to jump to any location in the memory. This instruction requires three bytes (two to specify the 16-bit address) with the low order address byte first followed by the high order address byte.



Source and Destination

TYPE OF ROTATE OR SHIFT	A	B	C	D	E	H	L	(HL)	(IX + d)	(IY + d)
'RLC'	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
'RRC'	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
'RL'	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
'RR'	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
'SLA'	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
'SRA'	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
'SRL'	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
'RLD'								ED 6F		
'RRD'								ED 67		

	A
RLCA	07
RRCA	0F
RLA	17
RRA	1F

ROTATES AND SHIFTS
TABLE 5.3–9

For example an unconditional Jump to memory location 3E32H would be:

Address A	C3	OP Code
A+1	32	Low order address
A+2	3E	High order address

The relative jump instruction uses only two bytes, the second byte is a signed two's complement displacement form the existing PC. This displacement can be in the range of +129 to -126 and is measured from the address of the instruction OP code.

Three types of register indirect jumps are also included. These instructions are implemented by loading the register pair HL or one of the index registers IX or IY directly into the PC. This capability allows for program jumps to be a function of previous calculations.

A call is a special form of a jump where the address of the byte following the call instruction is pushed onto the stack before the jump is made. A return instruction is the reverse of a call because the data on the top of the stack is popped directly into the PC to form a jump address. The call and return instructions allow for simple subroutine and interrupt handling. Two special return instructions have been included in the Z-80 family of components. The return from interrupt instruction (RETI) and the return from non maskable interrupt (RETN) are treated in the CPU as an unconditional return identical to the OP code C9H. The difference is that (RETI) can be used at the end of an interrupt routine and all Z-80 peripheral chips will recognize the execution of this instruction for proper control of nested priority interrupt handling. This instruction coupled with the Z-80 peripheral devices implementation simplifies the normal return from nested interrupt. Without this feature the following software sequence would be necessary to inform the interrupting device that the interrupt routine is completed:

		REGISTER ADDRESSING							REG. INDIR.	INDEXED	
	BIT	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
TEST 'BIT'	0	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
	1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
	2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
	3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
	4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
	5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
	6	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	GB 75	CB 76	DD CB d 76	FD CB d 76
	7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RESET BIT 'RES'	0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
	1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
	2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
	4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
	6	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET BIT 'SET'	0	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
	2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
	4	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
	6	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

BIT MANIPULATION GROUP
TABLE 5.3–10

Disable Interrupt	— prevent interrupt before routine is exited.
LD A, n OUT n, A	— notify peripheral that service routine is complete
Enable Interrupt	
Return	

This seven byte sequence can be replaced with the two byte RETI instruction in the Z-80. This is important since interrupt service time often must be minimized.

To facilitate program loop control the instruction DJNZ e can be used advantageously. This two byte, relative jump instruction decrements the B register and the jump occurs if the B register has not been decremented to zero. The relative displacement is expressed as a signed two's complement number. A simple example of its use might be:

Address	Instruction	Comments
N, N + 1	LD B, 7	; set B register to count of 7
N + 2 to N + 9	(Perform a sequence of instructions)	; loop to be performed 7 times
N + 10, N + 11	DJNZ -8	; to jump from N + 12 to N + 2
N + 12	(Next Instruction)	

CONDITION

			UN-COND.	CARRY	NON CARRY	ZERO	NON ZERO	PARITY EVEN	PARITY ODD	SIGN NEG	SIGN POS	REG B≠0
JUMP 'JP'	IMMED. EXT.	nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JUMP 'JR'	RELATIVE	PC+e	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
JUMP 'JP'	REG. INDIR.	(HL)	E9									
JUMP 'JP'		(IX)	DD E9									
JUMP 'JP'		(IY)	FD E9									
'CALL'	IMMED. EXT.	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DECREMENT B, JUMP IF NON ZERO 'DJNZ'	RELATIVE	PC+e										10 e-2
RETURN 'RET'	REGISTER INDIR.	(SP) (SP+1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETURN FROM INT 'RETI'	REG. INDIR.	(SP) (SP+1)	ED 4D									
RETURN FROM NON MASKABLE INT 'RETN'	REG. INDIR.	(SP) (SP+1)	ED 45									

NOTE—CERTAIN FLAGS HAVE MORE THAN ONE PURPOSE. REFER TO SECTION 6.0 FOR DETAILS

JUMP, CALL and RETURN GROUP
TABLE 5.3—11

Table 5.3-12 lists the eight OP codes for the restart instruction. This instruction is a single byte call to any of the eight addresses listed. The simple mnemonic for these eight calls is also shown. The value of this instruction is that frequently used routines can be called with this instruction to minimize memory usage.

CALL ADDRESS	OP CODE	
0000_H	C7	'RST 0'
0008_H	CF	'RST 8'
0010_H	D7	'RST 16'
0018_H	DF	'RST 24'
0020_H	E7	'RST 32'
0028_H	EF	'RST 40'
0030_H	F7	'RST 48'
0038_H	FF	'RST 56'

RESTART GROUP
TABLE 5.3–12

INPUT/OUTPUT

The Z-80 has an extensive set of Input and Output instructions as shown in table 5.3-13 and table 5.3-14. The addressing of the input or output device can be either absolute or register indirect, using the C register. Notice that in the register indirect addressing mode data can be transferred between the I/O devices and any of the internal registers. In addition eight block transfer instructions have been implemented. These instructions are similar to the memory block transfers except that they use register pair HL for a pointer to the memory source (output commands) or destination (input commands) while register B is used as a byte counter. Register C holds the address of the port for which the input or output command is desired. Since register B is eight bits in length, the I/O block transfer command handles up to 256 bytes.

In the instructions IN A, n and OUT n, A the I/O device address n appears in the lower half of the address bus (A_0-A_7) while the accumulator content is transferred in the upper half of the address bus. In all register indirect input output instructions, including block I/O transfers the content of register C is transferred to the lower half of the address bus (device address) while the content of register B is transferred to the upper half of the address bus.

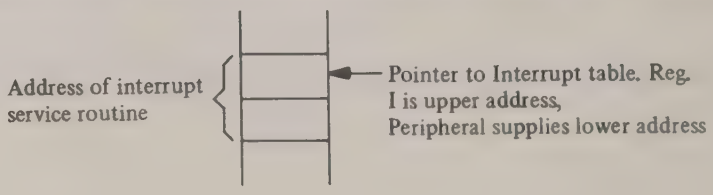
INPUT DESTINATION			SOURCE PORT ADDRESS: IMMED. (n)	SOURCE PORT ADDRESS: REG. INDIR. (C)
INPUT 'IN'	REG ADDRESSING	A	DB n	ED 78
		B		ED 40
		C		ED 48
		D		ED 50
		E		ED 58
		H		ED 60
		L		ED 68
'INI' – INPUT & Inc HL, Dec B	REG, INDIR	(HL)		ED A2
'INIR'– INP, Inc HL, Dec B, REPEAT IF B≠0				ED B2
'IND'– INPUT & Dec HL, Dec B				ED AA
'INDR'– INPUT, Dec HL, Dec B, REPEAT IF B≠0				ED BA

BLOCK INPUT COMMANDS

INPUT GROUP
TABLE 5.3–13

CPU CONTROL GROUP

The final table, table 5.3-15 illustrates the six general purpose CPU control instructions. The NOP is a do-nothing instruction. The HALT instruction suspends CPU operation until a subsequent interrupt is received, while the DI and EI are used to lock out and enable interrupts. The three interrupt mode commands set the CPU into any of the three available interrupt response modes as follows. If mode zero is set the interrupting device can insert any instruction on the data bus and allow the CPU to execute it. Mode 1 is a simplified mode where the CPU automatically executes a restart (RST) to location 0038H so that no external hardware is required. (The old PC content is pushed onto the stack). Mode 2 is the most powerful in that it allows for an indirect call to any location in memory. With this mode the CPU forms a 16-bit memory address where the upper 8-bits are the content of register I and the lower 8-bits are supplied by the interrupting device. This address points to the first of two sequential bytes in a table where the address of the service routine is located. The CPU automatically obtains the starting address and performs a CALL to this address.


Address of interrupt service routine
Pointer to Interrupt table. Reg. I is upper address, Peripheral supplies lower address

			SOURCE REGISTER A	B	C	D	E	H	L	REG. IND. (HL)
'OUT'	IMMED.	(n)	D3 n							
	REG. IND.	(C)	ED 79	ED 41	ED 49	ED 51	ED 59	ED 61	ED 69	
'OUTI' – OUTPUT Inc HL, Dec b	REG. IND.	(C)								ED A3
'OTIR' – OUTPUT, Inc HL, Dec B, REPEAT IF B≠0	REG. IND.	(C)								ED B3
'OUTD' – OUTPUT Dec HL & B	REG. IND.	(C)								ED AB
'OTDR' – OUTPUT, Dec HL & B, REPEAT IF B≠0	REG. IND.	(C)								ED BB

BLOCK OUTPUT COMMANDS

PORT DESTINATION ADDRESS

OUTPUT GROUP
TABLE 5.3–14

'NOP'	00	
'HALT'	76	
DISABLE INT '(DI)'	F3	
ENABLE INT '(EI)'	FB	
SET INT MODE 0 'IM0'	ED 46	8080A MODE
SET INT MODE 1 'IM1'	ED 56	CALL TO LOCATION 0038_H
SET INT MODE 2 'IM2'	ED 5E	INDIRECT CALL USING REGISTER I AND 8 BITS FROM INTERRUPTING DEVICE AS A POINTER.

MISCELLANEOUS CPU CONTROL
TABLE 5.3–15

6.0 FLAGS

Each of the two Z-80 CPU Flag registers contains six bits of information which are set or reset by various CPU operations. Four of these bits are testable; that is, they are used as conditions for jump, call or return instructions. For example a jump may be desired only if a specific bit in the flag register is set. The four testable flag bits are:

1) Carry Flag (C) – This flag is the carry from the highest order bit of the accumulator. For example, the carry flag will be set during an add instruction where a carry from the highest bit of the accumulator is generated. This flag is also set if a borrow is generated during a subtraction instruction. The shift and rotate instructions also affect this bit.

2) Zero Flag (Z) – This flag is set if the result of the operation loaded a zero into the accumulator. Otherwise it is reset.

3) Sign Flag (S) – This flag is intended to be used with signed numbers and it is set if the result of the operation was negative. Since bit 7 (MSB) represents the sign of the number (A negative number has a 1 in bit 7), this flag stores the state of bit 7 in the accumulator.

4) Parity/Overflow Flag (P/V) – This dual purpose flag indicates the parity of the result in the accumulator when logical operations are performed (such as AND A, B) and it represents overflow when signed two's complement arithmetic operations are performed. The Z-80 overflow flag indicates that the two's complement number in the accumulator is in error since it has exceeded the maximum possible (+127) or is less than the minimum possible (-128) number than can be represented in two's complement notation. For example consider adding:

```
+120 =    0111 1000
+105 =    0110 1001
-------  ----------
   C = 0  1110 0001  = -95 (wrong) Overflow has occured
```

Here the result is incorrect. Overflow has occurred and yet there is no carry to indicate an error. For this case the overflow flag would be set. Also consider the addition of two negative numbers:

```
 -5 =    1111 1011
-16 =    1111 0000
-------  ----------
   C = 1  1110 1011  = -21 correct
```

Notice that the answer is correct but the carry is set so that this flag can not be used as an overflow indicator. In this case the overflow would not be set.

For logical operations (AND, OR, XOR) this flag is set if the parity of the result is even and it is reset if it is odd.

There are also two non-testable bits in the flag register. Both of these are used for BCD arithmetic. They are:

1) Half carry (H) – This is the BCD carry or borrow result from the least significant four bits of operation. When using the DAA (Decimal Adjust Instruction) this flag is used to correct the result of a previous packed decimal add or subtract.

2) Subtract Flag (N) – Since the algorithm for correcting BCD operations is different for addition or subtraction, this flag is used to specify what type of instruction was executed last so that the DAA operation will be correct for either addition or subtraction.

The Flag register can be accessed by the programmer and its format is as follows:

S	Z	X	H	X	P/V	N	C

X means flag is indeterminate.

Table 6.0-1 lists how each flag bit is affected by various CPU instructions. In this table a '•' indicates that the instruction does not change the flag, an 'X' means that the flag goes to an indeterminate state, a '0' means that it is reset, a '1' means that it is set and the symbol '↕' indicates that it is set or reset according to the previous discussion. Note that any instruction not appearing in this table does not affect any of the flags.

Table 6.0-1 includes a few special cases that must be described for clarity. Notice that the block search instruction sets the Z flag if the last compare operation indicated a match between the source and the accumulator data. Also, the parity flag is set if the byte counter (register pair BC) is not equal to zero. This same use of the parity flag is made with the block move instructions. Another special case is during block input or output instructions, here the Z flag is used to indicate the state of register B which is used as a byte counter. Notice that when the I/O block transfer is complete, the zero flag will be reset to a zero (i.e. B=0) while in the case of a block move command the parity flag is reset when the operation is complete. A final case is when the refresh or I register is loaded into the accumulator, the interrupt enable flip flop is loaded into the parity flag so that the complete state of the CPU can be saved at any time.

Instruction	C	Z	P/V	S	N	H	Comments
ADD A, s; ADC A,s	↕	↕	V	↕	0	↕	8-bit add or add with carry
SUB s; SBC A, s, CP s, NEG	↕	↕	V	↕	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator
AND s	0	↕	P	↕	0	1	Logical operations
OR s; XOR s	0	↕	P	↕	0	0	And set's different flags
INC s	•	↕	V	↕	0	↕	8-bit increment
DEC m	•	↕	V	↕	1	↕	8-bit decrement
ADD DD, ss	↕	•	•	•	0	X	16-bit add
ADC HL, ss	↕	↕	V	↕	0	X	16-bit add with carry
SBC HL, ss	↕	↕	V	↕	1	X	16-bit subtract with carry
RLA; RLCA, RRA, RRCA	↕	•	•	•	0	0	Rotate accumulator
RL m; RLC m; RR m; RRC m SLA m; SRA m; SRL m	↕	↕	P	↕	0	0	Rotate and shift location s
RLD, RRD	•	↕	P	↕	0	0	Rotate digit left and right
DAA	↕	↕	P	↕	•	↕	Decimal adjust accumulator
CPL	•	•	•	•	1	1	Complement accumulator
SCF	1	•	•	•	0	0	Set carry
CCF	↕	•	•	•	0	X	Complement carry
IN r, (C)	•	↕	P	↕	0	0	Input register indirect
INI; IND; OUTI; OUTD	•	↕	X	X	1	X	Block input and output
INIR; INDR; OTIR; OTDR	•	1	X	X	1	X	Z = 0 if B ≠ 0 otherwise Z = 1
LDI, LDD	•	X	↕	X	0	0	Block transfer instructions
LDIR, LDDR	•	X	0	X	0	0	P/V = 1 if BC ≠ 0, otherwise P/V = 0
CPI, CPIR, CPD, CPDR	•	↕	↕	X	1	X	Block search instructions Z = 1 if A = (HL), otherwise Z = 0 P/V = 1 if BC ≠ 0, otherwise P/V = 0
LD A, I; LD A, R	•	↕	IFF	↕	0	0	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag
BIT b, s	•	↕	X	X	0	1	The state of bit b of location s is copied into the Z flag
NEG	↕	↕	V	↕	1	↕	Negate accumulator

The following notation is used in this table:

Symbol	Operation
C	Carry/link flag. C=1 if the operation produced a carry from the MSB of the operand or result.
Z	Zero flag. Z=1 if the result of the operation is zero.
S	Sign flag. S=1 if the MSB of the result is one.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V=1 if the result of the operation is even, P/V=0 if result is odd. If P/V holds overflow, P/V=1 if the result of the operation produced an overflow.
H	Half-carry flag. H=1 if the add or subtract operation produced a carry into or borrow from into bit 4 of the accumulator.
N	Add/Subtract flag. N=1 if the previous operation was a subtract.
	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
↕	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care."
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction. .
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range <0, 255>
nn	16-bit value in range <0, 65535>
m	Any 8-bit location for all the addressing modes allowed for the particular instruction.

SUMMARY OF FLAG OPERATION
TABLE 6.0-1

7.0 SUMMARY OF OP CODES AND EXECUTION TIMES

The following section gives a summary of the Z-80 instructions set. The instructions are logically arranged into groups as shown on tables 7.0-1 through 7.0-11. Each table shows the assembly language mnemonic OP code, the actual OP code, the symbolic operation, the content of the flag register following the execution of each instruction, the number of bytes required for each instruction as well as the number of memory cycles and the total number of T states (external clock periods) required for the fetching and execution of each instruction. Care has been taken to make each table self-explanatory without requiring any cross reference with the test or other tables.

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	OP-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T Cycles	Comments
LD r, r'	r ← r'	•	•	•	•	•	•	01 r r'	1	1	4	r, r' Reg.
LD r, n	r ← n	•	•	•	•	•	•	00 r 110	2	2	7	000 B
								← n →				001 C
LD r, (HL)	r ← (HL)	•	•	•	•	•	•	01 r 110	1	2	7	010 D
LD r, (IX+d)	r ← (IX+d)	•	•	•	•	•	•	11 011 101	3	5	19	011 E
								01 r 110				100 H
								← d →				101 L
LD r, (IY+d)	r ← (IY+d)	•	•	•	•	•	•	11 111 101	3	5	19	111 A
								01 r 110				
								← d →				
LD (HL), r	(HL) ← r	•	•	•	•	•	•	01 110 r	1	2	7	
LD (IX+d), r	(IX+d) ← r	•	•	•	•	•	•	11 011 101	3	5	19	
								01 110 r				
								← d →				
LD (IY+d), r	(IY+d) ← r	•	•	•	•	•	•	11 111 101	3	5	19	
								01 110 r				
								← d →				
LD (HL), n	(HL) ← n	•	•	•	•	•	•	00 110 110	2	3	10	
								← n →				
LD (IX+d), n	(IX+d) ← n	•	•	•	•	•	•	11 011 101	4	5	19	
								00 110 110				
								← d →				
								← n →				
LD (IY+d), n	(IY+d) ← n	•	•	•	•	•	•	11 111 101	4	5	19	
								00 110 110				
								← d →				
								← n →				
LD A, (BC)	A ← (BC)	•	•	•	•	•	•	00 001 010	1	2	7	
LD A, (DE)	A ← (DE)	•	•	•	•	•	•	00 011 010	1	2	7	
LD A, (nn)	A ← (nn)	•	•	•	•	•	•	00 111 010	3	4	13	
								← n →				
								← n →				
LD (BC), A	(BC) ← A	•	•	•	•	•	•	00 000 010	1	2	7	
LD (DE), A	(DE) ← A	•	•	•	•	•	•	00 010 010	1	2	7	
LD (nn), A	(nn) ← A	•	•	•	•	•	•	00 110 010	3	4	13	
								← n →				
								← n →				
LD A, I	A ← I	•	‡	IFF	‡	0	0	11 101 101	2	2	9	
								01 010 111				
LD A, R	A ← R	•	‡	IFF	‡	0	0	11 101 101	2	2	9	
								01 011 111				
LD I, A	I ← A	•	•	•	•	•	•	11 101 101	2	2	9	
								01 000 111				
LD R, A	R ← A	•	•	•	•	•	•	11 101 101	2	2	9	
								01 001 111				

Notes: r, r' means any of the registers A, B, C, D, E, H, L

IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,

‡ = flag is affected according to the result of the operation.

8-BIT LOAD GROUP
TABLE 7.0-1

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD dd, nn	dd ← nn	•	•	•	•	•	•	00 dd0 001 ← n → ← n →	3	3	10	dd Pair 00 BC 01 DE
LD IX, nn	IX ← nn	•	•	•	•	•	•	11 011 101 00 100 001 ← n → ← n →	4	4	14	10 HL 11 SP
LD IY, nn	IY ← nn	•	•	•	•	•	•	11 111 101 00 100 001 ← n → ← n →	4	4	14	
LD HL, (nn)	H ← (nn+1) L ← (nn)	•	•	•	•	•	•	00 101 010 ← n → ← n →	3	5	16	
LD dd, (nn)	dd_H ← (nn+1) dd_L ← (nn)	•	•	•	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	4	6	20	
LD IX, (nn)	IX_H ← (nn+1) IX_L ← (nn)	•	•	•	•	•	•	11 011 101 00 101 010 ← n → ← n →	4	6	20	
LD IY, (nn)	IY_H ← (nn+1) IY_L ← (nn)	•	•	•	•	•	•	11 111 101 00 101 010 ← n → ← n →	4	6	20	
LD (nn), HL	(nn+1) ← H (nn) ← L	•	•	•	•	•	•	00 100 010 ← n → ← n →	3	5	16	
LD (nn), dd	(nn+1) ← dd_H (nn) ← dd_L	•	•	•	•	•	•	11 101 101 01 dd0 011 ← n → ← n →	4	6	20	
LD (nn), IX	(nn+1) ← IX_H (nn) ← IX_L	•	•	•	•	•	•	11 011 101 00 100 010 ← n → ← n →	4	6	20	
LD (nn), IY	(nn+1) ← IY_H (nn) ← IY_L	•	•	•	•	•	•	11 111 101 00 100 010 ← n → ← n →	4	6	20	
LD SP, HL	SP ← HL	•	•	•	•	•	•	11 111 001	1	1	6	
LD SP, IX	SP ← IX	•	•	•	•	•	•	11 011 101 11 111 001	2	2	10	
LD SP, IY	SP ← IY	•	•	•	•	•	•	11 111 101 11 111 001	2	2	10	
PUSH qq	(SP-2) ← qq_L (SP-1) ← qq_H	•	•	•	•	•	•	11 qq0 101	1	3	11	qq Pair 00 BC 01 DE
PUSH IX	(SP-2) ← IX_L (SP-1) ← IX_H	•	•	•	•	•	•	11 011 101 11 100 101	2	4	15	10 HL 11 AF
PUSH IY	(SP-2) ← IY_L (SP-1) ← IY_H	•	•	•	•	•	•	11 111 101 11 100 101	2	4	15	
POP qq	qq_H ← (SP+1) qq_L ← (SP)	•	•	•	•	•	•	11 qq0 001	1	3	10	
POP IX	IX_H ← (SP+1) IX_L ← (SP)	•	•	•	•	•	•	11 011 101 11 100 001	2	4	14	
POP IY	IY_H ← (SP+1) IY_L ← (SP)	•	•	•	•	•	•	11 111 101 11 100 001	2	4	14	

Notes: dd is any of the register pairs BC, DE, HL, SP
qq is any of the register pairs AF, BC, DE, HL
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively.
E.g. BC_L = C, AF_H = A

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ flag is affected according to the result of the operation.

16-BIT LOAD GROUP
TABLE 7.0-2

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
EX DE, HL	DE ↔ HL	•	•	•	•	•	•	11 101 011	1	1	4	
EX AF, AF'	AF ↔ AF'	•	•	•	•	•	•	00 001 000	1	1	4	
EXX	(BC DE HL) ↔ (BC' DE' HL')	•	•	•	•	•	•	11 011 001	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔ (SP+1) L ↔ (SP)	•	•	•	•	•	•	11 100 011	1	5	19	
EX (SP), IX	IX_H ↔ (SP+1) IX_L ↔ (SP)	•	•	•	•	•	•	11 011 101 11 100 011	2	6	23	
EX (SP), IY	IY_H ↔ (SP+1) IY_L ↔ (SP)	•	•	•	•	•	•	11 111 101 11 100 011	2	6	23	
LDI	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1	•	•	↕ ①	•	0	0	11 101 101 10 100 000	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE+1 HL ← HL+1 BC ← BC-1 Repeat until BC = 0	•	•	0 ①	•	0	0	11 101 101 10 110 000	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
LDD	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1	•	•	↕ ①	•	0	0	11 101 101 10 101 000	2	4	16	
LDDR	(DE) ← (HL) DE ← DE-1 HL ← HL-1 BC ← BC-1 Repeat until BC = 0	•	•	0 ①	•	0	0	11 101 101 10 111 000	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A – (HL) HL ← HL+1 BC ← BC – 1	•	↕ ②	↕ ①	↕	1	↕	11 101 101 10 100 001	2	4	16	
CPIR	A – (HL) HL ← HL+1 BC ← BC-1 Repeat until A = (HL) or BC = 0	•	↕ ②	↕ ①	↕	1	↕	11 101 101 10 110 001	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A – (HL) HL ← HL-1 BC ← BC-1	•	↕ ②	↕ ①	↕	1	↕	11 101 101 10 101 001	2	4	16	
CPDR	A – (HL) HL ← HL-1 BC ← BC-1 Repeat until A = (HL) or BC = 0	•	↕ ②	↕ ①	↕	1	↕	11 101 101 10 111 001	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

Notes: ① P/V flag is 0 if the result of BC-1 = 0, otherwise P/V = 1
② Z flag is 1 if A = (HL), otherwise Z = 0.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
↕ = flag is affected according to the result of the operation.

EXCHANGE GROUP AND BLOCK TRANSFER AND SEARCH GROUP

TABLE 7.0-3

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD A, r	A ← A + r	‡	‡	V	‡	0	‡	10 [000] r	1	1	4	r Reg.
ADD A, n	A ← A + n	‡	‡	V	‡	0	‡	11 [000] 110 ← n →	2	2	7	000 B 001 C 010 D
ADD A, (HL)	A ← A + (HL)	‡	‡	V	‡	0	‡	10 [000] 110	1	2	7	011 E
ADD A, (IX+d)	A←A + (IX+d)	‡	‡	V	‡	0	‡	11 011 101 10 [000] 110 ← d →	3	5	19	100 H 101 L 111 A
ADD A, (IY+d)	A←A+(IY+d)	‡	‡	V	‡	0	‡	11 111 101 10 [000] 110 ← d →	3	5	19	
ADC A, s	A ← A + s + CY	‡	‡	V	‡	0	‡	[001]				s is any of r, n, (HL), (IX+d), (IY+d) as shown for ADD instruction
SUB s	A ← A − s	‡	‡	V	‡	1	‡	[010]				
SBC A, s	A ← A − s − CY	‡	‡	V	‡	1	‡	[011]				
AND s	A ← A ∧ s	0	‡	P	‡	0	1	[100]				
OR s	A ← A ∨ s	0	‡	P	‡	0	0	[110]				The indicated bits replace the 000 in the ADD set above
XOR s	A ← A ⊕ s	0	‡	P	‡	0	0	[101]				
CP s	A − s	‡	‡	V	‡	1	‡	[111]				
INC r	r ← r + 1	•	‡	V	‡	0	‡	00 r [100]	1	1	4	
INC (HL)	(HL) ← (HL)+1	•	‡	V	‡	0	‡	00 110 [100]	1	3	11	
INC (IX+d)	(IX+d) ← (IX+d)+1	•	‡	V	‡	0	‡	11 011 101 00 110 [100] ← d →	3	6	23	
INC (IY+d)	(IY+d) ← (IY+d) + 1	•	‡	V	‡	0	‡	11 111 101 00 110 [100] ← d ·	3	6	23	
DEC m	m←m−1	•	‡	V	‡	1	‡	[101]				m is any of r, (HL), (IX+d), (IY+d) as shown for INC Same format and states as INC Replace 100 with 101 in OP code

Notes: The V symbol in the r/V flag column indicates that the P/V flag contains the overflow of the result of the operation. Similarly the P symbol indicates parity. V = 1 means overflow, V = 0 means not overflow. P = 1 means parity of the result is even, P = 0 means parity of the result is odd

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown.
‡ = flag is affected according to the result of the operation

8-BIT ARITHMETIC AND LOGICAL GROUP
TABLE 7.0-4

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands	‡	‡	P	‡	•	‡	00 100 111	1	1	4	Decimal adjust accumulator
CPL	$A \leftarrow \overline{A}$	•	•	•	•	1	1	00 101 111	1	1	4	Complement accumulator (one's complement)
NEG	$A \leftarrow 0 - A$	‡	‡	V	‡	1	‡	11 101 101 01 000 100	2	2	8	Negate acc. (two's complement)
CCF	$CY \leftarrow \overline{CY}$	‡	•	•	•	0	X	00 111 111	1	1	4	Complement carry flag
SCF	$CY \leftarrow 1$	1	•	•	•	0	0	00 110 111	1	1	4	Set carry flag
NOP	No operation	•	•	•	•	•	•	00 000 000	1	1	4	
HALT	CPU halted	•	•	•	•	•	•	01 110 110	1	1	4	
DI	$IFF \leftarrow 0$	•	•	•	•	•	•	11 110 011	1	1	4	
EI	$IFF \leftarrow 1$	•	•	•	•	•	•	11 111 011	1	1	4	
IM 0	Set interrupt mode 0	•	•	•	•	•	•	11 101 101 01 000 110	2	2	8	
IM 1	Set interrupt mode 1	•	•	•	•	•	•	11 101 101 01 010 110	2	2	8	
IM2	Set interrupt mode 2	•	•	•	•	•	•	11 101 101 01 011 110	2	2	8	

Notes: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

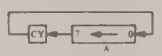
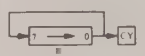
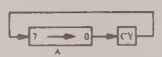
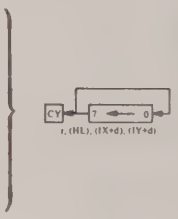
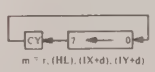
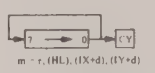
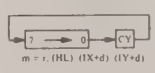
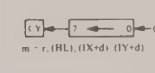
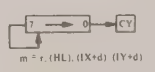
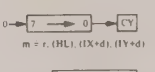
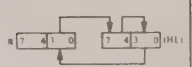
GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS
TABLE 7.0-5

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
ADD HL, ss	HL ← HL+ss	‡	•	•	•	0	X	00 ss1 001	1	3	11	ss Reg. 00 BC 01 DE 10 HL 11 SP
ADC HL, ss	HL←HL+ss+CY	‡	‡	V	‡	0	X	11 101 101 01 ss1 010	2	4	15	
SBC HL, ss	HL←HL-ss-CY	‡	‡	V	‡	1	X	11 101 101 01 ss0 010	2	4	15	
ADD IX, pp	IX ← IX + pp	‡	•	•	•	0	X	11 011 101 00 pp1 001	2	4	15	pp Reg. 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY←IY+rr	‡	•	•	•	0	X	11 111 101 00 rr1 001	2	4	15	rr Reg. 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss + 1	•	•	•	•	•	•	00 ss0 011	1	1	6	
INC IX	IX ← IX + 1	•	•	•	•	•	•	11 011 101 00 100 011	2	2	10	
INC IY	IY ← IY + 1	•	•	•	•	•	•	11 111 101 00 100 011	2	2	10	
DEC ss	ss ← ss - 1	•	•	•	•	•	•	00 ss1 011	1	1	6	
DEC IX	IX ← IX - 1	•	•	•	•	•	•	11 011 101 00 101 011	2	2	10	
DEC IY	IY ← IY - 1	•	•	•	•	•	•	11 111 101 00 101 011	2	2	10	

Notes: ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

16-BIT ARITHMETIC GROUP
TABLE 7.0-6

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
RLCA		‡	•	•	•	0	0	00 000 111	1	1	4	Rotate left circular accumulator
RLA		‡	•	•	•	0	0	00 010 111	1	1	4	Rotate left accumulator
RRCA		‡	•	•	•	0	0	00 001 111	1	1	4	Rotate right circular accumulator
RRA		‡	•	•	•	0	0	00 011 111	1	1	4	Rotate right accumulator
RLC r		‡	‡	P	‡	0	0	11 001 011 00 000 r	2	2	8	Rotate left circular register r
RLC (HL)		‡	‡	P	‡	0	0	11 001 011 00 000 110	2	4	15	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX+d)		‡	‡	P	‡	0	0	11 011 101 11 001 011 ← d → 00 000 110	4	6	23	
RLC (IY+d)		‡	‡	P	‡	0	0	11 111 101 11 001 011 ← d → 00 000 110	4	6	23	
RL m		‡	‡	P	‡	0	0	010				Instruction format and states are as shown for RLC,m. To form new OP-code replace 000 of RLC,m with shown code
RRC m		‡	‡	P	‡	0	0	001				
RR m		‡	‡	P	‡	0	0	011				
SLA m		‡	‡	P	‡	0	0	100				
SRA m		‡	‡	P	‡	0	0	101				
SRL m		‡	‡	P	‡	0	0	111				
RLD		•	‡	P	‡	0	0	11 101 101 01 101 111	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected
RRD		•	‡	P	‡	0	0	11 101 101 01 100 111	2	5	18	

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

ROTATE AND SHIFT GROUP
TABLE 7.0-7

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
BIT b, r	$Z \leftarrow \overline{r}_b$	•	‡	X	X	0	1	11 001 011 01 b r	2	2	8	r: 000 B, 001 C, 010 D, 011 E, 100 H, 101 L, 111 A
BIT b, (HL)	$Z \leftarrow \overline{(HL)}_b$	•	‡	X	X	0	1	11 001 011 01 b 110	2	3	12	b: 000 0, 001 1, 010 2, 011 3, 100 4, 101 5, 110 6, 111 7
BIT b, (IX+d)	$Z \leftarrow \overline{(IX+d)}_b$	•	‡	X	X	0	1	11 011 101 11 001 011 ← d → 01 b 110	4	5	20	
BIT b, (IY+d)	$Z \leftarrow \overline{(IY+d)}_b$	•	‡	X	X	0	1	11 111 101 11 001 011 ← d → 01 b 110	4	5	20	
SET b, r	$r_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 [11] b r	2	2	8	
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	•	•	•	•	11 001 011 [11] b 110	2	4	15	
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	•	•	•	•	•	•	11 011 101 11 001 011 ← d → [11] b 110	4	6	23	
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	•	•	•	•	•	•	11 111 101 11 001 011 ← d → [11] b 110	4	6	23	
RES b, m	$s_b \leftarrow 0$ m≡r, (HL), (IX+d), (IY+d)							[10]				To form new OP-code replace [11] of SET b,m with [10]. Flags and time states for SET instruction

Notes: The notation s_b indicates bit b (0 to 7) or location s.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

BIT SET, RESET AND TEST GROUP
TABLE 7.0-8

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP nn	PC ← nn	•	•	•	•	•	•	11 000 011 ← n → ← n →	3	3	10	
JP cc, nn	If condition cc is true PC ←nn, otherwise continue	•	•	•	•	•	•	11 cc 010 ← n → ← n →	3	3	10	
JR e	PC ← PC + e	•	•	•	•	•	•	00 011 000 ← e-2 →	2	3	12	
JR C, e	If C = 0, continue	•	•	•	•	•	•	00 111 000 ← e-2 →	2	2	7	If condition not met
	If C = 1, PC ← PC+e								2	3	12	If condition is met
JR NC, e	If C = 1, continue	•	•	•	•	•	•	00 110 000 ← e-2 →	2	2	7	If condition not met
	If C = 0, PC ← PC + e								2	3	12	If condition is met
JR Z, e	If Z = 0 continue	•	•	•	•	•	•	00 101 000 ← e-2 →	2	2	7	If condition not met
	If Z = 1, PC ← PC + e								2	3	12	If condition is met
JR NZ, e	If Z = 1, continue	•	•	•	•	•	•	00 100 000 ← e-2 →	2	2	7	If condition not met
	If Z = 0, PC ← PC + e								2	3	12	If condition met
JP (HL)	PC ← HL	•	•	•	•	•	•	11 101 001	1	1	4	
JP (IX)	PC ← IX	•	•	•	•	•	•	11 011 101 11 101 001	2	2	8	
JP (IY)	PC ← IY	•	•	•	•	•	•	11 111 101 11 101 001	2	2	8	
DJNZ,e	B ← B-1 If B = 0, continue	•	•	•	•	•	•	00 010 000 ← e-2 →	2	2	8	If B = 0
	If B ≠ 0, PC ← PC + e								2	3	13	If B ≠ 0

cc	Condition
000	NZ non zero
001	Z zero
010	NC non carry
011	C carry
100	PO parity odd
101	PE parity even
110	P sign positive
111	M sign negative

Notes: e represents the extension in the relative addressing mode.

e is a signed two's complement number in the range <-126, 129>

e-2 in the op-code provides an effective address of pc +e as PC is incremented by 2 prior to the addition of e.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

JUMP GROUP
TABLE 7.0-9

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC\leftarrow nn$	•	•	•	•	•	•	11 001 101 ← n → ← n →	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	•	•	•	•	11 cc 100 ← n → ← n →	3 3	3 5	10 17	If cc is false If cc is true
RET	$PC_L\leftarrow(SP)$ $PC_H\leftarrow(SP+1)$	•	•	•	•	•	•	11 001 001	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	•	•	•	•	11 cc 000	1 1	1 3	5 11	If cc is false If cc is true
RETI	Return from interrupt	•	•	•	•	•	•	11 101 101 01 001 101	2	4	14	
RETN	Return from non maskable interrupt	•	•	•	•	•	•	11 101 101 01 000 101	2	4	14	
RST p	$(SP-1)\leftarrow PC_H$ $(SP-2)\leftarrow PC_L$ $PC_H\leftarrow 0$ $PC_L\leftarrow P$	•	•	•	•	•	•	11 t 111	1	3	11	

cc	Condition	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign positive
111	M	sign negative

t	P
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown
‡ = flag is affected according to the result of the operation.

CALL AND RETURN GROUP
TABLE 7.0-10

Mnemonic	Symbolic Operation	Flags C	Z	P/V	S	N	H	Op-Code 76 543 210	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A, (n)	A ← (n)	•	▪	•	•	•	•	11 011 011 ← n →	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	•	‡	P	‡	0	‡	11 101 101 01 r 000	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL) ← (C) B ← B - 1 HL ← HL + 1	•	‡ ①	X	X	1	X	11 101 101 10 100 010	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL) ← (C) B ← B - 1 HL ← HL + 1 Repeat until B = 0	•	1	X	X	1	X	11 101 101 10 110 010	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
IND	(HL) ← (C) B ← B - 1 HL ← HL - 1	•	‡ ①	X	X	1	X	11 101 101 10 101 010	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INDR	(HL) ← (C) B ← B - 1 HL ← HL -1 Repeat until B = 0	•	1	X	X	1	X	11 101 101 10 111 010	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT (n), A	(n) ← A	•	•	•	•	•	•	11 010 011 ← n →	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	•	•	•	•	•	11 101 101 01 r 001	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	(C) ← (HL) B ← B - 1 HL ← HL + 1	•	‡ ①	X	X	1	X	11 101 101 10 100 011	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTIR	(C) ← (HL) B ← B - 1 HL ← HL + 1 Repeat until B = 0	•	1	X	X	1	X	11 101 101 10 110 011	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTD	(C) ← (HL) B ← B - 1 HL ← HL - 1	•	‡ ①	X	X	1	X	11 101 101 10 101 011	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTDR	(C) ← (HL) B ← B - 1 HL ← HL -1 Repeat until B = 0	•	1	X	X	1	X	11 101 101 10 111 011	2 2	5 (If B ≠ 0) 4 (If B = 0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

Notes: ① If the result of B - 1 is zero the Z flag is set, otherwise it is reset.

Flag Notation: • = flag not affected, 0 = flag reset, 1 = flag set, X = flag is unknown,
‡ = flag is affected according to the result of the operation.

INPUT AND OUTPUT GROUP
TABLE 7.0-11

8.0 INTERRUPT RESPONSE

The purpose of an interrupt is to allow peripheral devices to suspend CPU operation in an orderly manner and force the CPU to start a peripheral service routine. Usually this service routine is involved with the exchange of data, or status and control information, between the CPU and the peripheral. Once the service routine is completed, the CPU returns to the operation from which it was interrupted.

INTERRUPT ENABLE – DISABLE

The Z80 CPU has two interrupt inputs, a software maskable interrupt and a non maskable interrupt. The non maskable interrupt (NMI) can *not* be disabled by the programmer and it will be accepted whenever a peripheral device requests it. This interrupt is generally reserved for very important functions that must be serviced whenever they occur, such as an impending power failure. The maskable interrupt (INT) can be selectively enabled or disabled by the programmer. This allows the programmer to disable the interrupt during periods where his program has timing constraints that do not allow it to be interrupted. In the Z80 CPU there is an enable flip flop (called IFF) that is set or reset by the programmer using the Enable Interrupt (EI) and Disable Interrupt (DI) instructions. When the IFF is reset, an interrupt can not be accepted by the CPU.

Actually, for purposes that will be subsequently explained, there are two enable flip flops, called IFF_1 and IFF_2.

IFF_1	IFF_2
Actually disables interrupts from being accepted.	Temporary storage location for IFF_1.

The state of IFF_1 is used to actually inhibit interrupts while IFF_2 is used as a temporary storage location for IFF_1. The purpose of storing the IFF_1 will be subsequently explained.

A reset to the CPU will force both IFF_1 and IFF_2 to the reset state so that interrupts are disabled. They can then be enabled by an EI instruction at any time by the programmer. When an EI instruction is executed, any pending interrupt request will not be accepted until after the instruction following EI has been executed. This single instruction delay is necessary for cases when the following instruction is a return instruction and interrupts must not be allowed until the return has been completed. The EI instruction sets both IFF_1 and IFF_2 to the enable state. When an interrupt is accepted by the CPU, both IFF_1 and IFF_2 are automatically reset, inhibiting further interrupts until the programmer wishes to issue a new EI instruction. Note that for all of the previous cases, IFF_1 and IFF_2 are always equal.

The purpose of IFF_2 is to save the status of IFF_1 when a non maskable interrupt occurs. When a non maskable interrupt is accepted, IFF_1 is reset to prevent further interrupts until reenabled by the programmer. Thus, after a non maskable interrupt has been accepted, maskable interrupts are disabled but the previous state of IFF_1 has been saved so that the complete state of the CPU just prior to the non maskable interrupt can be restored at any time. When a Load Register A with Register I (LD A, I) instruction or a Load Register A with Register R (LD A, R) instruction is executed, the state of IFF_2 is copied into the parity flag where it can be tested or stored.

A second method of restoring the status of IFF_1 is thru the execution of a Return From Non Maskable Interrupt (RETN) instruction. Since this instruction indicates that the non maskable interrupt service routine is complete, the contents of IFF_2 are now copied back into IFF_1, so that the status of IFF_1 just prior to the acceptance of the non maskable interrupt will be restored automatically.

Figure 8.0-1 is a summary of the effect of different instructions on the two enable flip flops.

Action	IFF_1	IFF_2	
CPU Reset	0	0	
DI	0	0	
EI	1	1	
LD A, I	•	•	$IFF_2 \rightarrow$ Parity flag
LD A, R	•	•	$IFF_2 \rightarrow$ Parity flag
Accept NMI	0	•	
RETN	IFF_2	•	$IFF_2 \rightarrow IFF_1$

"•" indicates no change

FIGURE 8.0-1
INTERRUPT ENABLE/DISABLE FLIP FLOPS

CPU RESPONSE

Non Maskable

A nonmaskable interrupt will be accepted at all times by the CPU. When this occurs, the CPU ignores the next instruction that it fetches and instead does a restart to location 0066H. Thus, it behaves exactly as if it had received a restart instruction but, it is to a location that is not one of the 8 software restart locations. A restart is merely a call to a specific address in page 0 of memory.

Maskable

The CPU can be programmed to respond to the maskable interrupt in any one of three possible modes.

Mode 0

This mode is identical to the 8080A interrupt response mode. With this mode, the interrupting device can place any instruction on the data bus and the CPU will execute it. Thus, the interrupting device provides the next instruction to be executed instead of the memory. Often this will be a restart instruction since the interrupting device only need supply a single byte instruction. Alternatively, any other instruction such as a 3 byte call to any location in memory could be executed.

The number of clock cycles necessary to execute this instruction is 2 more than the normal number for the instruction. This occurs since the CPU automatically adds 2 wait states to an interrupt response cycle to allow sufficient time to implement an external daisy chain for priority control. Section 5.0 illustrates the detailed timing for an interrupt response. After the application of RESET the CPU will automatically enter interrupt Mode 0.

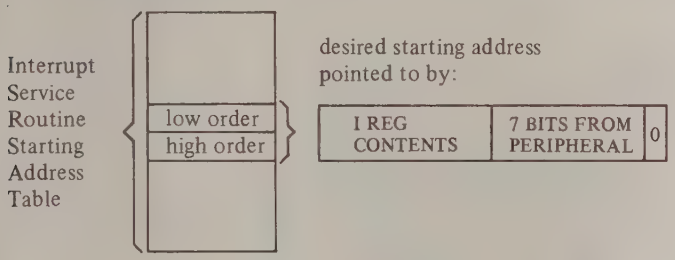
Mode 1

When this mode has been selected by the programmer, the CPU will respond to an interrupt by executing a restart to location 0038H. Thus the response is identical to that for a non maskable interrupt except that the call location is 0038H instead of 0066H. Another difference is that the number of cycles required to complete the restart instruction is 2 more than normal due to the two added wait states.

Mode 2

This mode is the most powerful interrupt response mode. With a single 8 bit byte from the user an indirect call can be made to any memory location.

With this mode the programmer maintains a table of 16 bit starting addresses for every interrupt service routine. This table may be located anywhere in memory. When an interrupt is accepted, a 16 bit pointer must be formed to obtain the desired interrupt service routine starting address from the table. The upper 8 bits of this pointer is formed from the contents of the I register. The I register must have been previously loaded with the desired value by the programmer, i.e. LD I, A. Note that a CPU reset clears the I register so that it is initialized to zero. The lower eight bits of the pointer must be supplied by the interrupting device. Actually, only 7 bits are required from the interrupting device as the least significant bit must be a zero. This is required since the pointer is used to get two adjacent bytes to form a complete 16 bit service routine starting address and the addresses must always start in even locations.



The first byte in the table is the least significant (low order) portion of the address. The programmer must obviously fill this table in with the desired addresses before any interrupts are to be accepted.

Note that this table can be changed at any time by the programmer (if it is stored in Read/Write Memory) to allow different peripherals to be serviced by different service routines.

Once the interrupting devices supplies the lower portion of the pointer, the CPU automatically pushes the program counter onto the stack, obtains the starting address from the table and does a jump to this address. This mode of response requires 19 clock periods to complete (7 to fetch the lower 8 bits from the interrupting device, 6 to save the program counter, and 6 to obtain the jump address.)

Note that the Z80 peripheral devices all include a daisy chain priority interrupt structure that automatically supplies the programmed vector to the CPU during interrupt acknowledge. Refer to the Z80-PIO, Z80-SIO and Z80-CTC manuals for details.

9.0 HARDWARE IMPLEMENTATION EXAMPLES

This chapter is intended to serve as a basic introduction to implementing systems with the Z80-CPU.

MINIMUM SYSTEM

Figure 9.0-1 is a diagram of a very simple Z-80 system. Any Z-80 system must include the following five elements:

1) Five volt power supply
2) Oscillator
3) Memory devices
4) I/O circuits
5) CPU

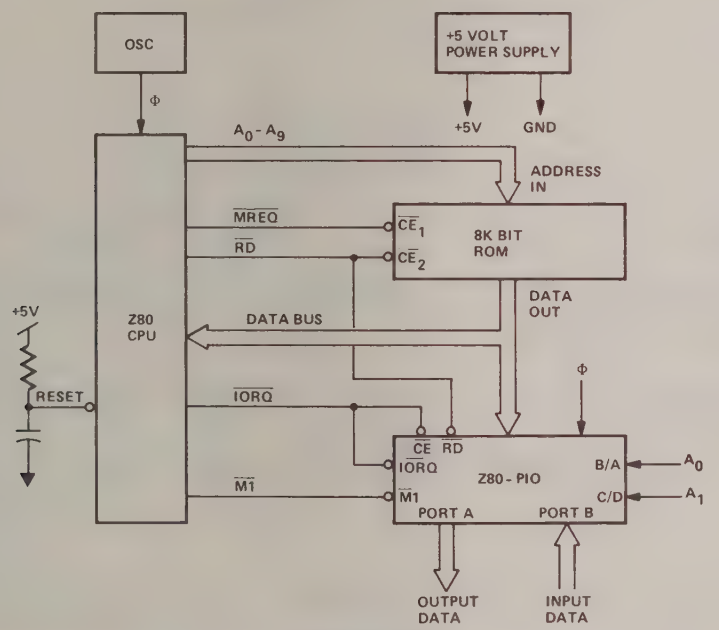


FIGURE 9.0-1
MINIMUM Z80 COMPUTER SYSTEM

Since the Z80-CPU only requires a single 5 volt supply, most small systems can be implemented using only this single supply.

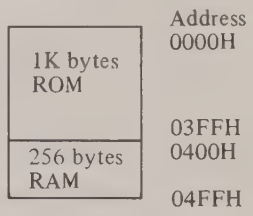
The oscillator can be very simple since the only requirement is that it be a 5 volt square wave. For systems not running at full speed, a simple RC oscillator can be used. When the CPU is operated near the highest possible frequency, a crystal oscillator is generally required because the system timing will not tolerate the drift or jitter that an RC network will generate. A crystal oscillator can be made from inverters and a few discrete components or monolithic circuits are widely available.

The external memory can be any mixture of standard RAM, ROM, or PROM. In this simple example we have shown a single 8K bit ROM (1K bytes) being utilized as the entire memory system. For this example we have assumed that the Z-80 internal register configuration contains sufficient Read/Write storage so that external RAM memory is not required.

Every computer system requires I/O circuits to allow it to interface to the "real world." In this simple example it is assumed that the output is an 8 bit control vector and the input is an 8 bit status word. The input data could be gated onto the data bus using any standard tri-state driver while the output data could be latched with any type of standard TTL latch. For this example we have used a Z80-PIO for the I/O circuit. This single circuit attaches to the data bus as shown and provides the required 16 bits of TTL compatible I/O. (Refer to the Z80-PIO manual for details on the operation of this circuit.) Notice in this example that with only three LSI circuits, a simple oscillator and a single 5 volt power supply, a powerful computer has been implemented.

ADDING RAM

Most computer systems require some amount of external Read/Write memory for data storage and to implement a "stack." Figure 9.0-2 illustrates how 256 bytes of static memory can be added to the previous example. In this example the memory space is assumed to be organized as follows:



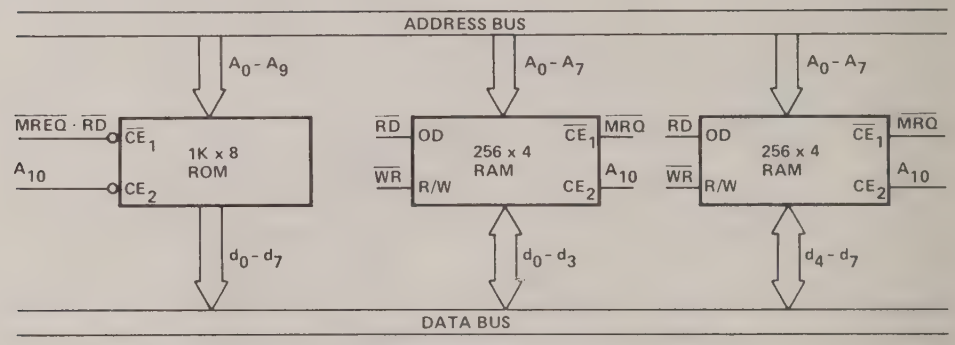


FIGURE 9.0-2
ROM & RAM IMPLEMENTATION EXAMPLE

In this diagram the address space is described in hexidecimal notation. For this example, address bit A_{10} separates the ROM space from the RAM space so that it can be used for the chip select function. For larger amounts of external ROM or RAM, a simple TTL decoder will be required to form the chip selects.

MEMORY SPEED CONTROL

For many applications, it may be desirable to use slow memories to reduce costs. The $\overline{\text{WAIT}}$ line on the CPU allows the Z-80 to operate with any speed memory. By referring back to section 4 you will notice that the memory access time requirements are most severe during the M1 cycle instruction fetch. All other memory accesses have an additional one half of a clock cycle to be completed. For this reason it may be desirable in some applications to add one wait state to the M1 cycle so that slower memories can be used. Figure 9.0-3 is an example of a simple circuit that will accomplish this task. This circuit can be changed to add a single wait state to any memory access as shown in Figure 9.0-4.

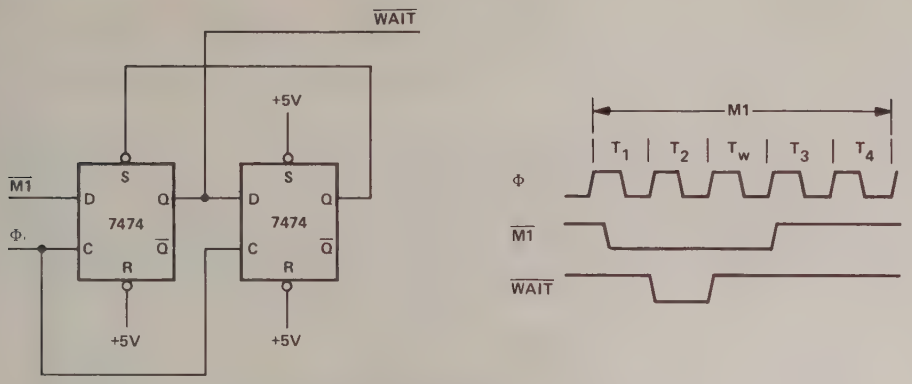


FIGURE 9.0-3
ADDING ONE WAIT STATE TO AN M1 CYCLE

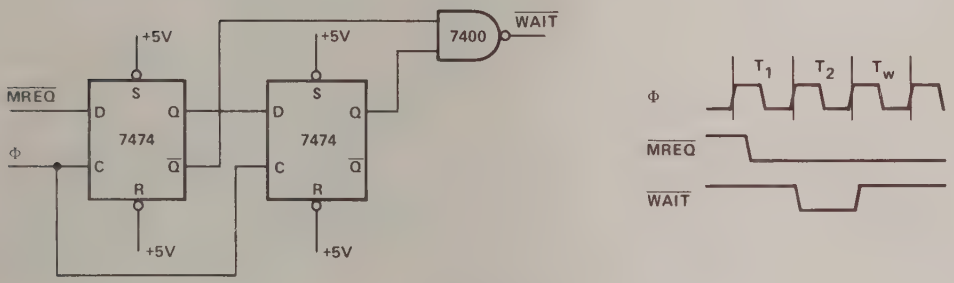


FIGURE 9.0-4
ADDING ONE WAIT STATE TO ANY MEMORY CYCLE

INTERFACING DYNAMIC MEMORIES

This section is intended only to serve as a brief introduction to interfacing dynamic memories. Each individual dynamic RAM has varying specifications that will require minor modifications to the description given here and no attempt will be made in this document to give details for any particular RAM. Separate application notes showing how the Z80-CPU can be interfaced to most popular dynamic RAM's are available from Zilog.

Figure 9.0-5 illustrates the logic necessary to interface 8K bytes of dynamic RAM using 18 pin 4K dynamic memories. This figure assumes that the RAM's are the only memory in the system so that A_{12} is used to select between the two pages of memory. During refresh time, all memories in the system must be read. The CPU provides the proper refresh address on lines A_0 through A_6. To add additional memory to the system it is necessary to only replace the two gates that operate on A_{12} with a decoder that operates on all required address bits. For larger systems, buffering for the address and data bus is also generally required.

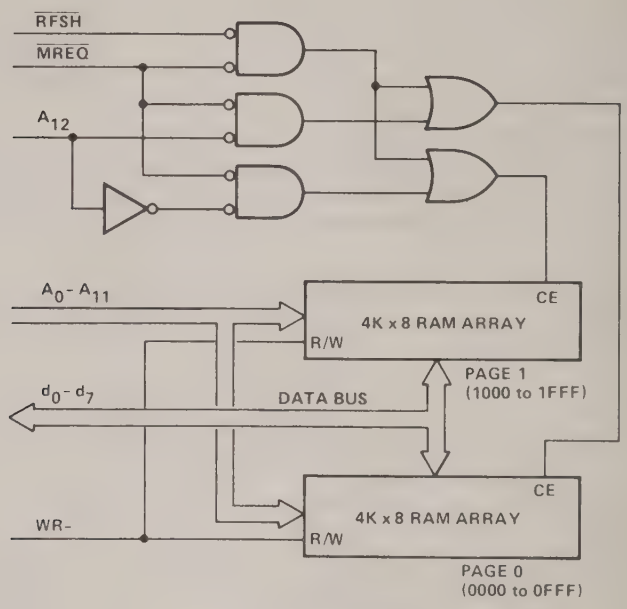


FIGURE 9.0-5
INTERFACING DYNAMIC RAMS

10.0 SOFTWARE IMPLEMENTATION EXAMPLES

10.1 METHODS OF SOFTWARE IMPLEMENTATION

Several different approaches are possible in developing software for the Z-80 (Figure 10.1). First of all, Assembly Language or PL/Z may be used as the source language. These languages may then be translated into machine language on a commercial time sharing facility using a cross-assembler or cross-compiler or, in the case of assembly language, the translation can be accomplished on a Z-80 Development System using a resident assembler. Finally, the resulting machine code can be debugged either on a time-sharing facility using a Z-80 simulator or on a Z-80 Development System which uses a Z80-CPU directly.

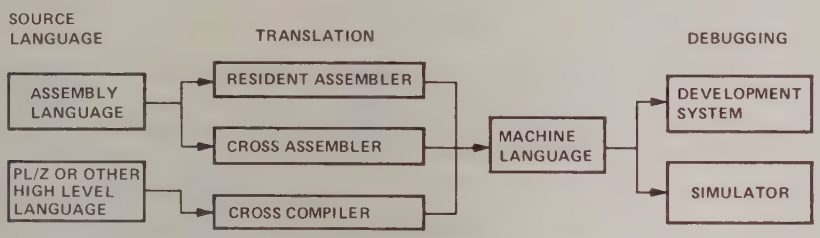


FIGURE 10.1

In selecting a source language, the primary factors to be considered are clarity and ease of programming vs. code efficiency. A high level language such as PL/Z with its machine independent constructs is typically better for formulating and maintaining algorithms, but the resulting machine code is usually somewhat less efficient than what can be written directly in assembly language. These tradeoffs can often be balanced by combining PL/Z and assembly language routines, identifying those portions of a task which must be optimized and writing them as assembly language subroutines.

Deciding whether to use a resident or cross assembler is a matter of availability and short-term vs. long-term expense. While the initial expenditure for a development system is higher than that for a time-sharing terminal, the cost of an individual assembly using a resident assembler is negligible while the same operation on a time-sharing system is relatively expensive and in a short time this cost can equal the total cost of a development system.

Debugging on a development system vs. a simulator is also a matter of availability and expense combined with operational fidelity and flexibility. As with the assembly process, debugging is less expensive on a development system than on a simulator available through time-sharing. In addition, the fidelity of the operating environment is preserved through real-time execution on a Z80-CPU and by connecting the I/O and memory components which will actually be used in the production system. The only advantage to the use of a simulator is the range of criteria which may be selected for such debugging procedures as tracing and setting breakpoints. This flexibility exists because a software simulation can achieve any degree of complexity in its interpretation of machine instructions while development system procedures have hardware limitations such as the capacity of the real-time storage module, the number of breakpoint registers and the pin configuration of the CPU. Despite such hardware limitations, debugging on a development system is typically more productive than on a simulator because of the direct interaction that is possible between the programmer and the authentic execution of his program.

10.2 SOFTWARE FEATURES OFFERED BY THE Z80-CPU

The Z-80 instruction set provides the user with a large and flexible repetoire of operations with which to formulate control of the Z80-CPU.

The primary, auxiliary and index registers can be used to hold the arguments of arithmetic and logical operations, or to form memory addresses, or as fast-access storage for frequently used data.

Information can be moved directly from register to register; from memory to memory; from memory to registers; or from registers to memory. In addition, register contents and register/memory contents can be exchanged without using temporary storage. In particular, the contents of primary and auxilary registers can be completely exchanged by executing only two instructions, EX and EXX. This register exchange procedure can be used to separate the set of working registers between different logical procedures or to expand the set of available registers in a single procedure.

Storage and retrieval of data between pairs of registers and memory can be controlled on a last-in first-out basis through PUSH and POP instructions which utilize a special stack pointer register, SP. This stack register is available both to manipulate data and to automatically store and retrieve addresses for subroutine linkage. When a subroutine is called, for example, the address following the CALL instruction is placed on the top of the push-down stack pointed to by SP. When a subroutine returns to the calling routine, the address on the top of the stack is used to set the program counter for the address of the next instruction. The stack pointer is adjusted automatically to reflect the current "top" stack position during PUSH, POP, CALL and RET instructions. This stack mechanism allows pushdown data stacks and subroutine calls to be nested to any practical depth because the stack area can potentially be as large as memory space.

The sequence of instruction execution can be controlled by six different flags (carry, zero, sign, parity/overflow, add-subtract, half-carry) which reflect the results of arithmetic, logical, shift and compare instructions. After the execution of an instruction which sets a flag, that flag can be used to control a conditional jump or return instruction. These instructions provide logical control following the manipulation of single bit, eight-bit byte (or) sixteen-bit data quantities.

A full set of logical operations, including AND, OR, XOR (exclusive - OR), CPL (NOR) and NEG (two's complement) are available for Boolean operations between the accumulator and 1) all other eight-bit registers, 2) memory locations or 3) immediate operands.

In addition, a full set of arithmetic and logical shifts in both directions are available which operate on the contents of all eight-bit primary registers or directly on any memory location. The carry flag can be included or simply set by these shift instructions to provide both the testing of shift results and to link register/register or register/memory shift operations.

10.3 EXAMPLES OF USE OF SPECIAL Z80 INSTRUCTIONS

A. Let us assume that a string of data in memory starting at location "DATA" is to be moved into another area of memory starting at location "BUFFER" and that the string length is 737 bytes. This operation can be accomplished as follows:

```
LD      HL , DATA       ; START ADDRESS OF DATA STRING
LD      DE , BUFFER     ; START ADDRESS OF TARGET BUFFER
LD      BC , 737        ; LENGTH OF DATA STRING
LDIR                    ; MOVE STRING - TRANSFER MEMORY POINTED TO
                        ; BY HL INTO MEMORY LOCATION POINTED TO BY DE
                        ; INCREMENT HL AND DE, DECREMENT BC
                        ; PROCESS UNTIL BC = 0.
```

11 bytes are required for this operation and each byte of data is moved in 21 clock cycles.

B. Let's assume that a string in memory starting at location "DATA" is to be moved into another area of memory starting at location "BUFFER" until an ASCII $ character (used as string delimiter) is found. Let's also assume that the maximum string length is 132 characters. The operation can be performed as follows:

```
        LD      HL , DATA       ; STARTING ADDRESS OF DATA STRING
        LD      DE , BUFFER     ; STA    NG ADDRESS OF TARGET BUFFER
        LD      BC , 132        ; MAXIMUM STRING LENGTH
        LD      A ,'$'          ; STRING DELIMITER CODE
LOOP:CP         (HL)            ; COMPARE MEMORY CONTENTS WITH DELIMITER
        JR      Z , END – $     ; GO TO END IF CHARACTERS EQUAL
        LDI                     ; MOVE CHARACTER (HL) to (DE)
                                ; INCREMENT HL AND DE, DECREMENT BC
        JP      PE , LOOP       ; GO TO "LOOP" IF MORE CHARACTERS
END:                            ; OTHERWISE, FALL THROUGH
                                ; NOTE: P/V FLAG IS USED
                                ; TO INDICATE THAT REGISTER BC WAS
                                ; DECREMENTED TO ZERO.
```

19 bytes are required for this operation.

C. Let us assume that a 16-digit decimal number represented in packed BCD format (two BCD digits/ byte) has to be shifted as shown in the Figure 10.2 in order to mechanize BCD multiplication or division. The operation can be accomplished as follows:

```
        LD      HL , DATA       ; ADDRESS OF FIRST BYTE
        LD      B , COUNT       ; SHIFT COUNT
        XOR     A               ; CLEAR ACCUMULATOR
ROTAT: RLD                      ; ROTATE LEFT LOW ORDER DIGIT IN ACC
                                ; WITH DIGITS IN (HL)
        INC     HL              ; ADVANCE MEMORY POINTER
        DJNZ    ROTAT – $       ; DECREMENT B AND GO TO ROTAT IF
                                ; B IS NOT ZERO, OTHERWISE FALL THROUGH
```

11 bytes are required for this operation.

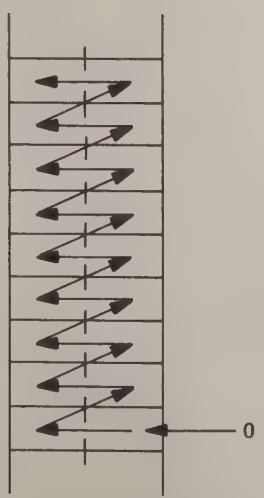


FIGURE 10.2

D. Let us assume that one number is to be subtracted from another and a) that they are both in packed BCD format, b) that they are of equal but varying length, and c) that the result is to be stored in the location of the minuend. The operation can be accomplished as follows:

```
        LD      HL , ARG1       ; ADDRESS OF MINUEND
        LD      DE , ARG2       ; ADDRESS OF SUBTRAHEND
        LD      B , LENGTH      ; LENGTH OF TWO ARGUMENTS
        AND     A               ; CLEAR CARRY FLAG
SUBDEC: LD      A , (DE)        ; SUBTRAHEND TO ACC
        SBC     A , (HL)        ; SUBTRACT (HL) FROM ACC
        DAA                     ; ADJUST RESULT TO DECIMAL CODED VALUE
        LD      (HL) , A        ; STORE RESULT
        INC     HL              ; ADVANCE MEMORY POINTERS
        INC     DE
        DJNZ    SUBDEC – $      ; DECREMENT B AND GO TO "SUBDEC" IF B
                                ; NOT ZERO, OTHERWISE FALL THROUGH
```

17 bytes are required for this operation.

10.4 EXAMPLES OF PROGRAMMING TASKS

A. The following program sorts an array of numbers each in the range ⟨0,255⟩ into ascending order using a standard exchange sorting algorithm.

```
01/22/76     11:14:37                BUBBLE LISTING                    PAGE 1
LOC     OBJ CODE    STMT    SOURCE STATEMENT

                      1  ;  *** STANDARD EXCHANGE (BUBBLE) SORT ROUTINE ***
                      2  ;
                      3  ;  AT ENTRY:  HL CONTAINS ADDRESS OF DATA
                      4  ;                C CONTAINS NUMBER OF ELEMENTS TO BE SORTED
                      5  ;                (1<C<=256)
                      6  ;
                      7  ;  AT EXIT: DATA SORTED IN ASCENDING ORDER
                      8  ;
                      9  ;  USE OF REGISTERS
                     10  ;
                     11  ;  REGISTER   CONTENTS
                     12  ;
                     13  ;  A          TEMPORARY STORAGE FOR CALCULATIONS
                     14  ;  B          COUNTER FOR DATA ARRAY
                     15  ;  C          LENGTH OF DATA ARRAY
                     16  ;  D          FIRST ELEMENT IN COMPARISON
                     17  ;  E          SECOND ELEMENT IN COMPARISON
                     18  ;  H          FLAG TO INDICATE EXCHANGE
                     19  ;  L          UNUSED
                     20  ;  IX         POINTER INTO DATA ARRAY
                     21  ;  IY         UNUSED
                     22  ;
0000    222600       23  SORT:   LD      (DATA), HL      ;SAVE DATA ADDRESS
0003    CB84         24  LOOP:   RES     FLAG, H         ;INITIALIZE EXCHANGE FLAG
0005    41           25          LD      B, C            ;INITIALIZE LENGTH COUNTER
0006    05           26          DEC     B               ;ADJUST FOR TESTING
0007    DD2A2600     27          LD      IX, (DATA)      ;INITIALIZE ARRAY POINTER
000B    DD7E00       28  NEXT:   LD      A, (IX)'        ;FIRST ELEMENT IN COMPARISON
000E    57           29          LD      D, A            ;TEMPORARY STORAGE FOR ELEMENT
000F    DD5E01       30          LD      E, (IX+1)       ;SECOND ELEMENT IN COMPARISON
0012    93           31          SUB     E               ;COMPARISON FIRST TO SECOND
0013    3008         32          JR      NC,  NOEX-$     ;IF FIRST > SECOND, NO JUMP
0015    DD7300       33          LD      (IX), E         ;EXCHANGE ARRAY ELEMENTS
0018    DD7201       34          LD      (IX+1), D
001B    CBC4         35          SET     FLAG, H         ;RECORD EXCHANGE OCCURRED
001D    DD23         36  NOEX:   INC     IX              ;POINT TO NEXT DATA ELEMENT
001F    10EA         37          DJNZ    NEXT-$          ;COUNT NUMBER OF COMPARISONS
                     38                                  ;REPEAT IF MORE DATA PAIRS
0021    CB44         39          BIT     FLAG, H         ;DETERMINE IF EXCHANGE OCCURRED
0023    20DE         40          JR      NZ, LOOP-$      ;CONTINUE IF DATA UNSORTED
0025    C9           41          RET                     ;OTHERWISE, EXIT
                     42  ;
0026                 43  FLAG:   EQU     0               ;DESIGNATION OF FLAG BIT
0026                 44  DATA:   DEFS    2               ;STORAGE FOR DATA ADDRESS
                     45          END
```

B. The following program multiplies two unsigned 16 bit integers and leaves the result in the HL register pair.

```
01/22/76     11:32:36                MULTIPLY LISTING                PAGE 1
LOC    OBJ CODE    STMT    SOURCE STATEMENT

0000                 1    MULT::   UNSIGNED SIXTEEN BIT INTEGER MULTIPLY.
                     2    ;        ON ENTRANCE:  MULTIPLIER IN DE.
                     3    ;                      MULTIPLICAND IN HL.
                     4    ;
                     5    ;        ON EXIT:  RESULT IN HL.
                     6    ;
                     7    ;        REGISTER USES:
                     8    ;
                     9    ;
                    10    ;        H      HIGH ORDER PARTIAL RESULT
                    11    ;        L      LOW ORDER PARTIAL RESULT
                    12    ;        D      HIGH ORDER MULTIPLICAND
                    13    ;        E      LOW ORDER MULTIPLICAND
                    14    ;        B      COUNTER FOR NUMBER OF SHIFTS
                    15    ;        C      HIGH ORDER BITS OF MULTIPLIER
                    16    ;        A      LOW ORDER BITS OF MULTIPLIER
                    17    ;
0000   0610         18             LD     B, 16;        NUMBER OF BITS- INITIALIZE
0002   4A           19             LD     C, D;         MOVE MULTIPLIER
0003   7B           20             LD     A, E;
0004   EB           21             EX     DE, HL;       MOVE MULTIPLICAND
0005   210000       22             LD     HL, 0;        CLEAR PARTIAL RESULT
0008   CB39         23    MLOOP:   SRL    C;            SHIFT MULTIPLIER RIGHT
000A   1F           24             RRA                  LEAST SIGNIFICANT BIT IS
                    25    ;                             IN CARRY.
000B   3001         26             JR     NC, NOADD-$;  IF NO CARRY, SKIP THE ADD.
000D   19           27             ADD    HL, DE;       ELSE ADD MULTIPLICAND TO
                    28    ;                             PARTIAL RESULT.
000E   EB           29    NOADD:   EX     DE, HL;       SHIFT MULTIPLICAND LEFT
000F   29           30             ADD    HL, HL;       BY MULTIPLYING IT BY TWO.
0010   EB           31             EX     DE, HL;
0011   10F5         32             DJNZ   MLOOP-$;      REPEAT UNTIL NO MORE BITS.
0013   C9           33             RET;
                    34             END;
```

HEATHKIT

Absolute Maximum Ratings

Temperature Under Bias	Specified operating range.
Storage Temperature	-65°C to +150°C
Voltage On Any Pin with Respect to Ground	-0.3V to +7V
Power Dissipation	1.5W

*Comment

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Note: For Z80-CPU all AC and DC characteristics remain the same for the military grade parts except I_{cc}.

I_{cc} = 200 mA

Z80-CPU D.C. Characteristics

T_A = 0°C to 70°C, V_{cc} = 5V ± 5% unless otherwise specified

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{cc} -.6		V_{cc}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{cc}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250μA
I_{CC}	Power Supply Current			150	mA	
I_{LI}	Input Leakage Current			10	μA	V_{IN}=0 to V_{cc}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT}=2.4 to V_{cc}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leq V_{IN} \leq V_{cc}$

Capacitance

T_A = 25°C, f = 1 MHz,
unmeasured pins returned to ground

Symbol	Parameter	Max.	Unit
C_Φ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

Z80-CPU Ordering Information

C – Ceramic
P – Plastic
S – Standard 5V ±5% 0° to 70°C
E – Extended 5V ±5% -40° to 85°C
M – Military 5V ±10% -55° to 125°C

Z80A-CPU D.C. Characteristics

T_A = 0°C to 70°C, V_{cc} = 5V ± 5% unless otherwise specified

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{cc} -.6		V_{cc}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{cc}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250μA
I_{CC}	Power Supply Current		90	200	mA	
I_{LI}	Input Leakage Current			10	μA	V_{IN}=0 to V_{cc}
I_{LOH}	Tri-State Output Leakage Current in Float			10	μA	V_{OUT}=2.4 to V_{cc}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	μA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	μA	$0 \leq V_{IN} \leq V_{cc}$

Capacitance

T_A = 25°C, f = 1 MHz,
unmeasured pins returned to ground

Symbol	Parameter	Max	Unit
C_Φ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

Z80A-CPU Ordering Information

C – Ceramic
P – Plastic
S – Standard 5V ±5% 0° to 70°C

A.C. Characteristics — Z80-CPU

T_A = 0°C to 70°C, Vcc = +5V ± 5%, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
Φ	t_c	Clock Period	4	[12]	μsec	
	t_w (ΦH)	Clock Pulse Width, Clock High	180	[E]	nsec	
	t_w (ΦL)	Clock Pulse Width, Clock Low	180	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		145	nsec	C_L = 50pF
	$t_{F(AD)}$	Delay to Float		110	nsec	
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable from $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec	
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		230	nsec	C_L = 50pF
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	60		nsec	
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{WR}$	[7]			
	t_H	Any Hold Time for Setup Time	0		nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low		100	nsec	C_L = 50pF
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		100	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		100	nsec	
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		90	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		110	nsec	
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		100	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		110	nsec	
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		100	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		130	nsec	
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		100	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		110	nsec	
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		80	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		90	nsec	
	$t_{DH\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		100	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{WR}$ Low	[10]		nsec	
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ Low		130	nsec	C_L = 50pF
	$t_{DH(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		130	nsec	
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		180	nsec	C_L = 50pF
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ High		150	nsec	
$\overline{WAIT}$	$t_{s(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
$\overline{INT}$	$t_{s(IT)}$	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{NMI}$	$t_{w(\overline{NML})}$	Pulse Width, $\overline{NMI}$ Low	80		nsec	
$\overline{BUSRQ}$	$t_{s(BQ)}$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay From Rising Edge of Clock, $\overline{BUSAK}$ Low		120	nsec	C_L = 50pF
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay From Falling Edge of Clock, $\overline{BUSAK}$ High		110	nsec	
$\overline{RESET}$	$t_{s(RS)}$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	90		nsec	
	$t_{F(C)}$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		100	nsec	
	t_{mr}	$\overline{M1}$ Stable Prior to $\overline{IORQ}$ (Interrupt Ack.)	[11]		nsec	

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 75$

[2] $t_{aci} = t_c - 80$

[3] $t_{ca} = t_{w(\Phi L)} + t_r - 40$

[4] $t_{caf} = t_{w(\Phi L)} + t_r - 60$

[5] $t_{dcm} = t_c - 210$

[6] $t_{dci} = t_{w(\Phi L)} + t_r - 210$

[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 80$

[8] $t_{w(MRL)} = t_c - 40$

[9] $t_{w(MRH)} = t_{w(\Phi H)} + t_f - 30$

[10] $t_{w(\overline{WRL})} = t_c - 40$

[11] $t_{mr} = 2t_c + t_{w(\Phi H)} + t_f - 80$

NOTES

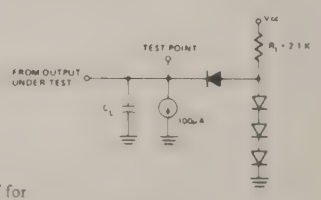
A Data should be enabled onto the CPU data bus when $\overline{RD}$ is active. During interrupt acknowledge data should be enabled when $\overline{M1}$ and $\overline{IORQ}$ are both active.

B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

C The $\overline{RESET}$ signal must be active for a minimum of 3 clock cycles.

D Output Delay vs. Loaded Capacitance
TA = 70°C Vcc = +5V ±5%

Add 10nsec delay for each 50pf increase in load up to a maximum of 200pf for the data bus & 100pf for address & control lines

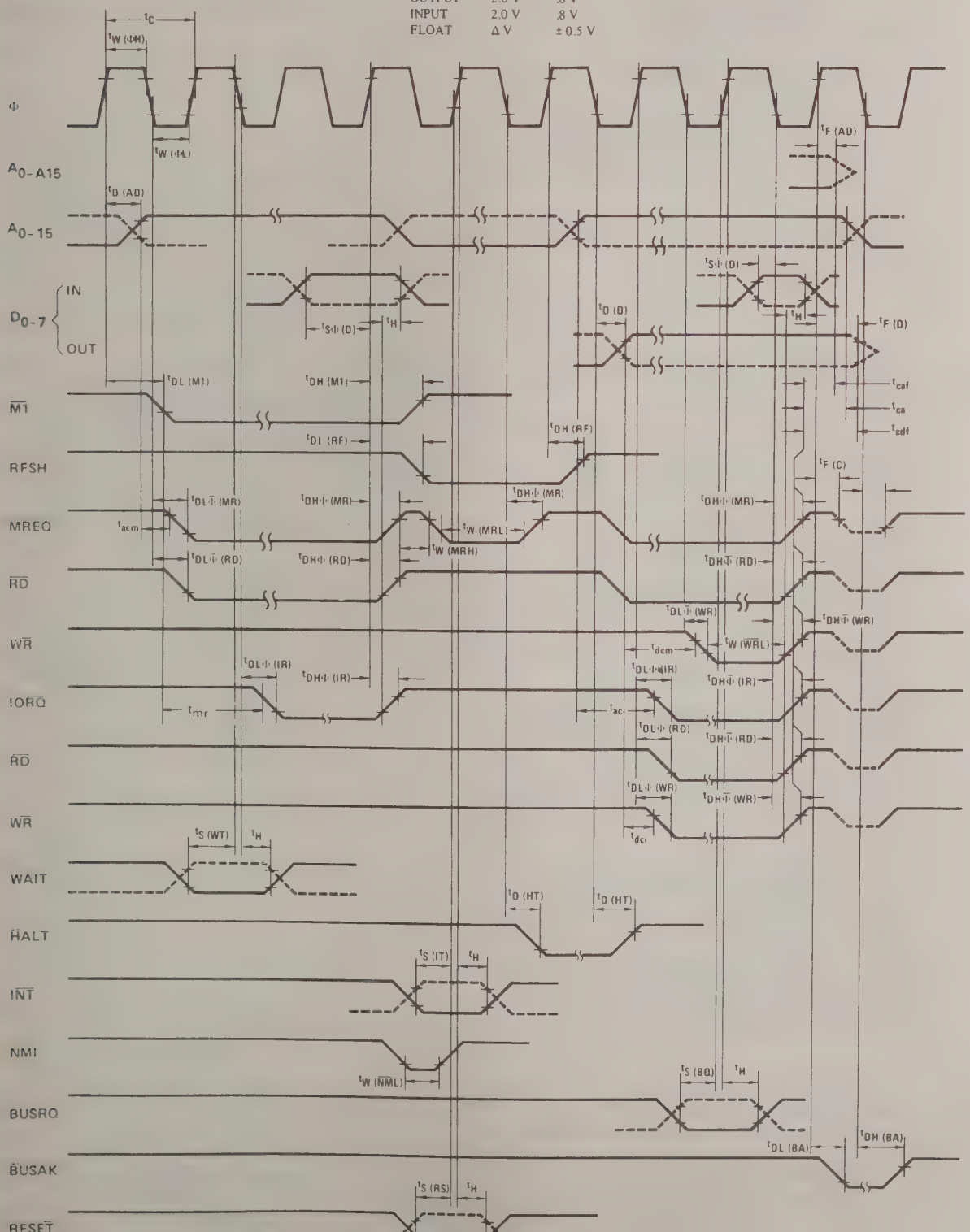


Load circuit for Output

A.C. Timing Diagram

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	V_{cc} -.6V	.45V
OUTPUT	2.0 V	.8 V
INPUT	2.0 V	.8 V
FLOAT	Δ V	± 0.5 V



A.C. Characteristics — Z80A-CPU

T_A = 0°C to 70°C, Vcc = +5V ± 5%, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
Φ	t_c	Clock Period	25	[12]	μsec	
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	110	[E]	nsec	
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	110	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		110	nsec	C_L = 50pF
	$t_{F(AD)}$	Delay to Float		90	nsec	
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable from $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec	
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		150	nsec	C_L = 50pF
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	35		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	50		nsec	
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{WR}$	[7]			
	t_H	Any Hold Time for Setup Time		0	nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low		85	nsec	C_L = 50pF
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		85	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		85	nsec	
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		75	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		85	nsec	
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		85	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		85	nsec	
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		85	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		95	nsec	
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		85	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		85	nsec	
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		65	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		80	nsec	
	$t_{DH\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		80	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{WR}$ Low	[10]		nsec	
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ Low		100	nsec	C_L = 50pF
	$t_{DH(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		100	nsec	
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		130	nsec	C_L = 50pF
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ High		120	nsec	
$\overline{WAIT}$	$t_{s(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	C_L 50pF
$\overline{INT}$	$t_{s(IT)}$	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{NMI}$	$t_{w(\overline{NML})}$	Pulse Width, $\overline{NMI}$ Low	80		nsec	
$\overline{BUSRQ}$	$t_{s(BQ)}$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	50		nsec	
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay From Rising Edge of Clock, $\overline{BUSAK}$ Low		100	nsec	C_L = 50pF
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay From Falling Edge of Clock, $\overline{BUSAK}$ High		100	nsec	
$\overline{RESET}$	$t_{s(RS)}$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	60		nsec	
	$t_{F(C)}$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		80	nsec	
	t_{mr}	M1 Stable Prior to IORQ (Interrupt Ack.)	[11]		nsec	

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 65$

[2] $t_{aci} = t_c - 70$

[3] $t_{ca} = t_{w(\Phi L)} + t_r - 50$

[4] $t_{caf} = t_{w(\Phi L)} + t_r - 45$

[5] $t_{dcm} = t_c - 170$

[6] $t_{dci} = t_{w(\Phi L)} + t_r - 170$

[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 70$

[8] $t_{w(\overline{MRL})} = t_c - 30$

[9] $t_{w(\overline{MRH})} = t_{w(\Phi H)} + t_f - 20$

[10] $t_{w(\overline{WRL})} = t_c - 30$

[11] $t_{mr} = 2t_c + t_{w(\Phi H)} + t_f - 65$

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

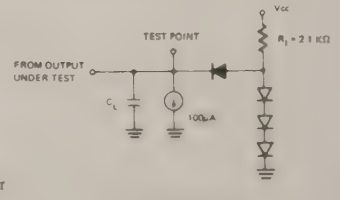
NOTES

A. Data should be enabled onto the CPU data bus when $\overline{RD}$ is active. During interrupt acknowledge data should be enabled when $\overline{M1}$ and $\overline{IORQ}$ are both active.

B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

C. The $\overline{RESET}$ signal must be active for a minimum of 3 clock cycles.

D. Output Delay vs. Loaded Capacitance
TA = 70°C Vcc = +5V ±5%
Add 10nsec delay for each 50pf increase in load up to maximum of 200pf for data bus and 100pf for address & control lines.



Load circuit for Output

12.0

Zilog Z80-CPU INSTRUCTION SET

ADC HL, ss	Add with Carry Reg. pair ss to HL
ADC A, s	Add with carry operand s to Acc.
ADD A, n	Add value n to Acc.
ADD A, r	Add Reg. r to Acc.
ADD A, (HL)	Add location (HL) to Acc.
ADD A, (IX+d)	Add location (IX+d) to Acc.
ADD A, (IY+d)	Add location (IY+d) to Acc.
ADD HL, ss	Add Reg. pair ss to HL
ADD IX, pp	Add Reg. pair pp to IX
ADD IY, rr	Add Reg. pair rr to IY
AND s	Logical 'AND' of operand s and Acc.
BIT b, (HL)	Test BIT b of location (HL)
BIT b, (IX+d)	Test BIT b of location (IX+d)
BIT b, (IY+d)	Test BIT b of location (IY+d)
BIT b, r	Test BIT b of Reg. r
CALL cc, nn	Call subroutine at location nn if condition cc if true
CALL nn	Unconditional call subroutine at location nn
CCF	Complement carry flag
CP s	Compare operand s with Acc.
CPD	Compare location (HL) and Acc. decrement HL and BC
CPDR	Compare location (HL) and Acc. decrement HL and BC, repeat until BC=0
CPI	Compare location (HL) and Acc. increment HL and decrement BC
CPIR	Compare location (HL) and Acc. increment HL, decrement BC repeat until BC=0
CPL	Complement Acc. (1's comp)
DAA	Decimal adjust Acc.
DEC m	Decrement operand m
DEC IX	Decrement IX
DEC IY	Decrement IY
DEC ss	Decrement Reg. pair ss
DI	Disable interrupts
DJNZ e	Decrement B and Jump relative if B≠0
EI	Enable interrupts
EX (SP), HL	Exchange the location (SP) and HL
EX (SP), IX	Exchange the location (SP) and IX
EX (SP), IY	Exchange the location (SP) and IY
EX AF, AF'	Exchange the contents of AF and AF'
EX DE, HL	Exchange the contents of DE and HL
EXX	Exchange the contents of BC, DE, HL with contents of BC', DE', HL' respectively
HALT	HALT (wait for interrupt or reset)
IM 0	Set interrupt mode 0
IM 1	Set interrupt mode 1
IM 2	Set interrupt mode 2
IN A, (n)	Load the Acc. with input from device n
IN r, (C)	Load the Reg. r with input from device (C)
INC (HL)	Increment location (HL)
INC IX	Increment IX
INC (IX+d)	Increment location (IX+d)
INC IY	Increment IY
INC (IY+d)	Increment location (IY+d)
INC r	Increment Reg. r
INC ss	Increment Reg. pair ss
IND	Load location (HL) with input from port (C), decrement HL and B
INDR	Load location (HL) with input from port (C), decrement HL and decrement B, repeat until B=0
INI	Load location (HL) with input from port (C); and increment HL and decrement B

INIR	Load location (HL) with input from port (C), increment HL and decrement B, repeat until B=0
JP (HL)	Unconditional Jump to (HL)
JP (IX)	Unconditional Jump to (IX)
JP (IY)	Unconditonal Jump to (IY)
JP cc, nn	Jump to location nn if condition cc is true
JP nn	Unconditional jump to location nn
JP C, e	Jump relative to PC+e if carry=1
JR e	Unconditional Jump relative to PC+e
JP NC, e	Jump relative to PC+e if carry=0
JR NZ, e	Jump relative to PC+e if non zero (Z=0)
JR Z, e	Jump relative to PC+e if zero (Z=1)
LD A, (BC)	Load Acc. with location (BC)
LD A, (DE)	Load Acc. with location (DE)
LD A, I	Load Acc. with I
LD A, (nn)	Load Acc. with location nn
LD A, R	Load Acc. with Reg. R
LD (BC), A	Load location (BC) with Acc.
LD (DE), A	Load location (DE) with Acc.
LD (HL), n	Load location (HL) with value n
LD dd, nn	Load Reg. pair dd with value nn
LD HL, (nn)	Load HL with location (nn)
LD (HL), r	Load location (HL) with Reg. r
LD I, A	Load I with Acc.
LF IX, nn	Load IX with value nn
LD IX, (nn)	Load IX with location (nn)
LD (IX+d), n	Load location (IX+d) with value n
LD (IX+d), r	Load location (IX+d) with Reg. r
LD IY, nn	Load IY with value nn
LD IY, (nn)	Load IY with location (nn)
LD (IY+d), n	Load location (IY+d) with value n
LD (IY+d), r	Load location (IY+d) with Reg. r
LD (nn), A	Load location (nn) with Acc.
LD (nn), dd	Load location (nn) with Reg. pair dd
LD (nn), HL	Load location (nn) with HL
LD (nn), IX	Load location (nn) with IX
LD (nn), IY	Load location (nn) with IY
LD R, A	Load R with Acc.
LD r, (HL)	Load Reg. r with location (HL)
LD r, (IX+d)	Load Reg. r with location (IX+d)
LD r, (IY+d)	Load Reg. r with location (IY+d)
LD r, n	Load Reg. r with value n
LD r, r'	Load Reg. r with Reg. r'
LD SP, HL	Load SP with HL
LD SP, IX	Load SP with IX
LD SP, IY	Load SP with IY
LDD	Load location (DE) with location (HL), decrement DE, HL and BC
LDDR	Load location (DE) with location (HL), decrement DE, HL and BC; repeat until BC=0
LDI	Load location (DE) with location (HL), increment DE, HL, decrement BC
LDIR	Load location (DE) with location (HL), increment DE, HL, decrement BC and repeat until BC=0
NEG	Negate Acc. (2's complement)
NOP	No operation
OR s	Logical 'OR' or operand s and Acc.
OTDR	Load output port (C) with location (HL) decrement HL and B, repeat until B=0
OTIR	Load output port (C) with location (HL), increment HL, decrement B, repeat until B=0
OUT (C), r	Load output port (C) with Reg. r
OUT (n), A	Load output port (n) with Acc.
OUTD	Load output port (C) with location (HL), decrement HL and B
OUTI	Load output port (C) with location (HL), increment HL and decrement B

POP IX	Load IX with top of stack
POP IY	Load IY with top of stack
POP qq	Load Reg. pair qq with top of stack
PUSH IX	Load IX onto stack
PUSH IY	Load IY onto stack
PUSH qq	Load Reg. pair qq onto stack
RES b, m	Reset Bit b of operand m
RET	Return from subroutine
RET cc	Return from subroutine if condition cc is true
RETI	Return from interrupt
RETN	Return from non maskable interrupt
RL m	Rotate left through carry operand m
RLA	Rotate left Acc. through carry
RLC (HL)	Rotate location (HL) left circular
RLC (IX+d)	Rotate location (IX+d) left circular
RLC (IY+d)	Rotate location (IY+d) left circular
RLC r	Rotate Reg. r left circular
RLCA	Rotate left circular Acc.
RLD	Rotate digit left and right between Acc. and location (HL)
RR m	Rotate right through carry operand m
RRA	Rotate right Acc. through carry
RRC m	Rotate operand m right circular
RRCA	Rotate right circular Acc.
RRD	Rotate digit right and left between Acc. and location (HL)
RST p	Restart to location p
SBC A, s	Subtract operand s from Acc. with carry
SBC HL, ss	Subtract Reg. pair ss from HL with carry
SCF	Set carry flag (C=1)
SET b, (HL)	Set Bit b of location (HL)
SET b, (IX+d)	Set Bit b of location (IX+d)
SET b, (IY+d)	Set Bit b of location (IY+d)
SET b, r	Set Bit b of Reg. r
SLA m	Shift operand m left arithmetic
SRA m	Shift operand m right arithmetic
SRL m	Shift operand m right logical
SUB s	Subtract operand s from Acc.
XOR s	Exclusive 'OR' operand s and Acc.

INS8250 ASYNCHRONOUS COMMUNICATIONS ELEMENT*

INS8250 Functional Pin Description

The function of all INS8250 input/output pins are described in the following paragraphs. (See the INS8250 Block Diagram, Illustration Booklet, Page 16). Some of these descriptions reference internal circuits. A low in these descriptions represents a logic 0 (0 volt nominal) and a high represents a logic 1 (+2.4 volts nominal).

INPUT SIGNALS

Chip Select (CSØ, CS1, $\overline{CS2}$), Pins 12 - 14: When CSØ and CS1 are high and $\overline{CS2}$ is low, the chip is selected. Chip selection is complete when the decoded chip select signal is latched with an active (low) Address Strobe ($\overline{ADS}$) input. This enables communication between the INS8250 and the CPU.

Data Input Strobe (DISTR, $\overline{DISTR}$), Pins 22 and 21: When DISTR is high or $\overline{DISTR}$ is low while the chip is selected, this allows the CPU to read status information or data from a selected register of the INS8250.

NOTE: Only an active DISTR or $\overline{DISTR}$ input is required to transfer data from the INS8250 during a read operation. Therefore, tie either the DISTR input permanently low or the $\overline{DISTR}$ input permanently high, if not used.

Data Output Strobe (DOSTR, $\overline{DOSTR}$), Pins 19 and 18: When DOSTR is high or $\overline{DOSTR}$ is low while the chip is selected, this allows the CPU to write data or control words into a selected register of the INS8250.

NOTE: Only an active DOSTR or $\overline{DOSTR}$ input is required to transfer data to the INS8250 during a write operation. Therefore, tie either the DOSTR input permanently low or the $\overline{DOSTR}$ input permanently high, if not used.

Address Strobe ($\overline{ADS}$), Pin 25: When low, it provides latching for the Register Select (A0, A1, A2) and Chip Select (CSØ, CS1, $\overline{CS2}$) signals.

NOTE: An active $\overline{ADS}$ input is required when the Register Select (A0, A1, A2) signals are not stable for the duration of a read or write operation. If not required, tie the $\overline{ADS}$ input permanently low.

Register Select (A0, A1, A2), Pins 26 - 28: These three inputs are used during a read or write operation to select an INS8250 register to read from or write into as indicated in the table below. Note that the state of the Divisor Latch Access Bit (DLAB), which is the most significant bit of the line control register, affects the selection of certain INS8250 registers. The DLAB is reset low when the Master Reset (MR) input is active (low); the DLAB must be set high by the system software to access the baud generator divisor latches.

DLAB	A_2	A_1	A_0	Register
0	0	0	0	Receiver Buffer (read), Transmitter Holding Register (write)
0	0	0	1	Interrupt Enable
X	0	1	0	Interrupt Identification (read only)
X	0	1	1	Line Control
X	1	0	0	MODEM Control
X	1	0	1	Line Status
X	1	1	0	MODEM Status
X	1	1	1	None
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

Master Reset (MR), Pin 35: When high, it clears all the registers (except the receiver buffer, transmitter holding, and divisor latches), and the control logic of the INS8250. Also, the state of various output signals (SOUT, INTRPT, $\overline{OUT\ 1}$, $\overline{OUT\ 2}$, $\overline{RTS}$, $\overline{DTR}$) are affected by an active MR input. (Refer to Table 1 on Page 14-3.)

Receiver Clock (RCLK), Pin 9: This input is the 16x baud rate clock for the receiver section of the chip.

Serial Input (SIN), Pin 10: Serial data input from the communications link (peripheral device, MODEM, or data set).

* Portions of this section are reprinted with the permission of National Semiconductor.

Clear to Send ($\overline{CTS}$), Pin 36: The $\overline{CTS}$ signal is a MODEM control function input whose condition can be tested by the CPU by reading bit 4 (CTS) of the MODEM status register. Bit 0 (DCTS) of the MODEM status register indicates whether the $\overline{CTS}$ input has changed state since the previous reading of the MODEM status register.

NOTE: Whenever the CTS bit of the MODEM status register changes state, an interrupt is generated if enabled.

Data Set Ready ($\overline{DSR}$), Pin 37: When low, it indicates that the MODEM or data set is ready to establish the communications link and transfer data with the INS8250. The $\overline{DSR}$ signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 5 (DSR) of the MODEM status register. Bit 1 (DDSR) of the MODEM status register indicates whether the $\overline{DSR}$ input has changed state since the previous reading of the MODEM status register.

NOTE: Whenever the DSR bit of the MODEM status register changes state, an interrupt is generated if enabled.

Received Line Signal Detect ($\overline{RLSD}$), Pin 38: When low, it indicates that the data carrier has been detected by the MODEM or data set. The $\overline{RLSD}$ signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 7 (RLSD) of the MODEM status register. Bit 3 (DRLSD) of the MODEM status register indicates whether the $\overline{RLSD}$ input has changed state since the previous reading of the MODEM status register.

NOTE: Whenever the RLSD bit of the MODEM status register changes state, an interrupt is generated if enabled.

Ring Indicator ($\overline{RI}$), Pin 39: When low, it indicates that a telephone ringing signal has been received by the MODEM or data set. The $\overline{RI}$ signal is a MODEM control function input whose condition can be tested by the CPU by reading bit 6 (RI) of the MODEM status register. Bit 2 (TERI) of the MODEM status register indicates whether the $\overline{RI}$ input has changed from a low to a high state since the previous reading of the MODEM status register.

NOTE: Whenever the RI bit of the MODEM status register changes from a high to a low state, an interrupt is generated if enabled.

V_{CC}, Pin 40: +5-volt supply.

V_{SS}, Pin 20: Ground (0-volt) reference.

OUTPUT SIGNALS

Data Terminal Ready ($\overline{DTR}$), Pin 33: When low, it informs the MODEM or data set that the INS8250 is ready to communicate. The DTR output signal can be set to an active low by programming bit 0 (DTR) of the MODEM control register to a high level. The $\overline{DTR}$ signal is set high upon a Master Reset operation.

Request to Send ($\overline{RTS}$), Pin 32: When low, it informs the MODEM or data set that the INS8250 is ready to transmit data. The $\overline{RTS}$ output signal can be set to an active low by programming bit 1 (RTS) of the MODEM control register. The $\overline{RTS}$ signal is set high upon a Master Reset operation.

Output 1 ($\overline{OUT\ 1}$), Pin 34: A user-designated output that can be set to an active low by programming bit 2 ($\overline{OUT\ 1}$) of the MODEM control register to a high level. The $\overline{OUT\ 1}$ signal is set high upon a Master Reset operation.

Output 2 ($\overline{OUT\ 2}$), Pin 31: A user-designated output that can be set to an active low by programming bit 3 (OUT 2) of the MODEM control register to a high level. The $\overline{OUT\ 2}$ signal is set high upon a Master Reset operation.

Chip Select Out (CSOUT), Pin 24: When high, it indicates that the chip has been selected by active CSØ, CS1, and $\overline{CS2}$ inputs. No data transfer can be initiated until the CSOUT signal is a logic 1.

Driver Disable (DDIS), Pin 23: Goes low whenever the CPU is reading data from the INS8250. A high-level DDIS output can be used to disable an external transceiver (if used between the CPU and INS8250 on the D_7 - D_0 Data Bus) at all times, except when the CPU is reading data.

Baud Out ($\overline{BAUDOUT}$), Pin 15: 16x clock signal for the transmitter section of the INS8250 . The clock rate is equal to the main reference oscillator frequency divided by the specified divisor in the baud generator divisor latches. The $\overline{BAUDOUT}$ may also be used for the receiver section by typing this output to the RCLK input of the chip.

Interrupt (INTRPT), Pin 30: Goes high whenever any one of the following interrupt sources has an active high condition: Receiver Error Flag; Received Data Available; Transmitter Holding Register Empty; and MODEM Status. The INTRPT signal is reset low upon a Master Reset operation.

Serial Output (SOUT), Pin 11: Composite serial data output to the communications link (peripheral, MODEM or data set). The SOUT signal is set to the Marking (logic 1) state upon a Master Reset operation.

INPUT/OUTPUT SIGNALS

Data (D_7 - D_0) Bus, Pins 1 - 8: This bus comprises eight TRI-STATE input/output lines. The bus provides bidirectional communications between the INS8250 and the CPU. Data, control words, and status information are transferred via the D_7 - D_0 data bus.

External Clock Input/Output (XTAL 1, XTAL 2), Pins 16 and 17: These two pins connect the main timing reference (crystal or signal clock) to the INS8250.

Register/Signal	Reset Control	Reset State
Receiver Buffer Register	First Word Received	Data
Transmitter Holding Register	Writing into the Transmitter Holding Register	Data
Interrupt Enable Register	Master Reset	All bits Low (0 - 3 forced and 4 - 7 permanent)
Interrupt Identification Register	Master Reset	Bit 0 is High and Bits 1 - 7 Are Permanently Low
Line Control Register	Master Reset	All Bits Low
MODEM Control Register	Master Reset	All Bits Low
Line Status Register	Master Reset	All Bits Low, Except Bits 5 & 6 Are High
MODEM Status Register	Master Reset	Bits 0 - 3 Low
	MODEM Signal Inputs	Bits 4 - 7 — Input Signal
Divisor Latch (low order bits)	Writing into the Latch	Data
Divisor Latch (high order bits)	Writing into the Latch	Data
SOUT	Master Reset	High
$\overline{\text{BAUDOUT}}$	Writing into Either Divisor Latch	Low
CSOUT	$\overline{\text{ADS}}$ Strobe Signal and State of Chip Select Lines	High/Low
DDIS	DDIS = $\overline{\text{CSOUT} \cdot \text{RCLK} \cdot \text{DISTR}}$ (AT Master Reset, the CPU sets RCLK and DISTR low.)	High
INTRPT	Master Reset	Low
$\overline{\text{OUT 2}}$	Master Reset	High
$\overline{\text{RTS}}$	Master Reset	High
$\overline{\text{DTR}}$	Master Reset	High
$\overline{\text{OUT 1}}$	Master Reset	High
D_7 - D_0 Data Bus Lines	In TRI-STATE Mode, Unless CSOUT • DISTR = High or CSOUT • DOSTR = High	TRI-STATE DATA (ACE to CPU) DATA (CPU to ACE)

Table 1

Reset Control of Registers and Pinout Signals.

Programming

When you use Heath software, you will not be concerned with programming the 8250 ACE (asynchronous communications element) in the H88-3. However, this section will be indispensable if you intend to assemble your own program code.

In order to easily program the 8250, you should:

1. Disable all UART interrupts by clearing the interrupt enable register.
2. Set the ACE in its loop-back mode.
3. Program the ACE as you want it.
4. Read a character.
5. Wait two character times.
6. Read a second character.
7. Take the ACE out of the loop-back mode.

INS8250 ACCESSIBLE REGISTERS

You (the system programmer) may access or control any of the INS8250 registers summarized in Table 1 via the CPU. These registers are used to control INS8250 operations and to transmit and receive data.

INS8250 Line Control Register

Specify the format of the asynchronous data communications exchange via the Line Control Register. In addition to controlling the format, you may retrieve the contents of the Line Control Register for inspection. This feature simplifies system programming and eliminates the need for separate storage in system memory of the line characteristics. The contents of the Line Control Register are indicated in Table 2 and are described below.

Bits 0 and 1: These two bits specify the number of bits in each transmitted or received serial character. The encoding of bits 0 and 1 is as follows:

Bit 1	Bit 0	Word Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

Bit 2: This bit specifies the number of Stop bits in each transmitted or received serial character. If bit 2 is a logic 0, 1 Stop bit is generated or checked in the transmit or receive data, respectively. If bit 2 is a logic 1 when a 5-bit word length is selected via bits 0 and 1, 1-1/2 Stop bits are generated or checked. If bit 2 is a logic 1 when either a 6-, 7-, or 8-bit word length is selected, 2 Stop bits are generated or checked.

Bit 3: This is the Parity Enable bit. When Bit 3 is a logic 1, a Parity bit is generated (transmit data) or checked (receive data) between the last data word bit and Stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data word bits and the Parity bit are summed.)

Bit 4: This is the Even Parity Select bit. When bit 3 is a logic 1 and bit 4 is a logic 0, an odd number of logic 1s is transmitted or checked in the data word bits and Parity bit. When bit 3 is a logic 1 and bit 4 is a logic 1, an even number of bits is transmitted or checked.

Bit 5: This is the Stick Parity bit. When bit 3 is a logic 1 and bit 5 is a logic 1, the Parity bit is transmitted and then detected by the receiver in the opposite state indicated by bit 4.

Bit 6: This is the Set Break Control bit. When bit 6 is a logic 1, the serial output (SOUT) is forced to the spacing (logic 0) state and remains there (until reset by a low-level bit 6) regardless of other transmitter activity. This feature enables the CPU to alert a terminal in a computer communications system.

Bit 7: This is the Divisor Latch Access bit (DLAB). It must be set high (logic 1) to access the Divisor Latches of the baud rate generator during a Read or Write operation. It must be set low (logic 0) to access the receiver buffer, the transmitter holding register, or the interrupt enable register.

Bit No.	Register Address									
	0 DLAB = 0	0 DLAB = 0	1 DLAB = 0	2	3	4	5	6	0 DLAB = 1	1 DLAB = 1
	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Identification Register	Line Control Register	MODEM Control Register	Line Status Register	MODEM Status Register	Divisor Latch (LS)	Divisor Latch (MS)
0	Data Bit 0*	Data Bit 0	Enable Received Data Available Interrupt (ERBFI)	"0" if Interrupt Pending	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (DCTS)	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Enable Transmitter Holding Register Empty Interrupt (ETBEI)	Interrupt ID Bit (0)	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OR)	Delta Data Set Ready (DDSR)	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Enable Receiver Line Status Interrupt (ELSI)	Interrupt ID Bit (1)	Number of Stop Bits (STB)	Out 1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	Enable MODEM Status Interrupt (EDSSI)	0	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Receive Line Signal Detect (DRLSD)	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Stick Parity	0	Transmitter Holding Register Empty (THRE)	Data Set Ready (DSR)	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	0	Set Break	0	Transmitter Shift Register Empty (TSRE)	Ring Indicator (RI)	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	0	Divisor Latch Access Bit (DLAB)	0	0	Received Line Signal Detect (RLSD)	Bit 7	Bit 15

* Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

Table 2
Summary of INS8250 Accessible Registers.

8250 PROGRAMMABLE BAUD RATE GENERATOR

The 8250 contains a programmable baud rate generator that takes the 1.8432 MHz clock and divides it by any divisor from 1 to 2^{16} - 1. The output frequency of the baud generator is 16 × the baud rate. Two 8-bit latches store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization in order to insure desired operation of the baud rate generator. Upon loading either of the divisor latches, a 16-bit baud counter is immediately loaded. This prevents long counts on initial load.

Table 3 illustrates the standard baud rates and the contents of the LS (least significant) and MS (most significant) latches expressed in byte octal.

BAUD RATE	DIVISOR LATCH (LS)	(MS)
75	000	006
110	027	004
134.5	131	003
150	000	003
300	200	001
600	300	000
1200	140	000
2400	060	000
4800	030	000
9600	014	000
19200	006	000

Table 3
Baud Rates.

LINE STATUS REGISTER

This 8-bit register provides status information to the CPU concerning the data transfer. The contents of the line status register are indicated in Table 2 and are described below.

Bit 0: This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to a logic 1 whenever a complete incoming character has been received and transferred into the receiver buffer register. Bit 0 may be reset to a logic 0 either by the CPU reading the data in the receiver buffer register or by writing a logic 0 into it from the CPU.

Bit 1: This bit is the Overrun Error (OE) indicator. Bit 1 indicates that data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the receiver buffer register, thereby destroying the previous character. The OE indicator is reset whenever the CPU reads the contents of the line status register.

Bit 2: This bit is the Parity Error (PE) indicator. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even-parity-select bit. The PE bit is set to a logic 1 upon detection of a parity error and is reset to a logic 0 whenever the CPU reads the contents of the line status register.

Bit 3: This bit is the Framing Error (FE) indicator. Bit 3 indicates that the received character did not have a valid Stop bit. Bit 3 is set to a logic 1 whenever the Stop bit following the last data bit or parity bit is detected as a zero bit (Spacing level).

Bit 4: This bit is the Break Interrupt (BI) indicator. Bit 4 is set to a logic 1 whenever the received data input is held in the Spacing (logic 0) state for longer than a full word transmission time (that is, the total time of Start bit + data bits + Parity + Stop bits).

NOTE: Bits 1 through 4 are the error conditions that produce a Receiver Line Status interrupt whenever any of the corresponding conditions are detected.

Bit 5: This bit is the Transmitter Holding Register Empty (THRE) indicator. Bit 5 indicates that the INS8250 is ready to accept a new character for transmission. In addition, this bit causes the INS8250 to issue an interrupt to the CPU when the Transmit Holding Register Empty Interrupt enable is set high. The THRE bit is set to a logic 1 when a character is transferred from the transmitter holding register into the transmitter shift register. The bit is reset to logic 0 concurrently with the loading of the transmitter holding register by the CPU.

Bit 6: This bit is the Transmitter Shift Register Empty (TSRE) indicator. Bit 6 is set to a logic 1 whenever the transmitter shift register is idle. It is reset to logic 0 upon a data transfer from the transmitter holding register to the transmitter shift register. Bit 6 is a read-only bit.

Bit 7: This bit is permanently set to logic 0.

INTERRUPT IDENTIFICATION REGISTER

The INS8250 has an on-chip interrupt capability that allows for complete flexibility in interfacing to all the popular microprocessors presently available. In order to provide minimum software overhead during data character transfers, the INS8250 prioritizes interrupts into four levels. The four levels of interrupt conditions are as follows: Receiver Line Status (priority 1); Received Data Ready (priority 2); Transmitter Holding Register Empty (priority 3); and MODEM Status (priority 4).

Information indicating that a prioritized interrupt is pending and the source of that interrupt are stored in the interrupt identification register (refer to Table 4). The interrupt identification register (IIR, when addressed during chip-select time, freezes the highest priority interrupt pending and no other interrupts are acknowledged until the particular interrupt is serviced by the CPU. The contents of the IIR are indicated in Table 2 and are described below.

Bit 0: This bit can be used in either a hardwired prioritized or polled environment to indicate whether an interrupt is pending. When bit 0 is a logic 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is a logic 1, no interrupt is pending and polling (if used) continues.

Bits 1 and 2: These two bits of the IIR are used to identify the highest priority interrupt pending as indicated in table 3.

Bits 3 through 7: These five bits of the IIR are always logic 0.

INTERRUPT ENABLE REGISTER

This 8-bit register enables the four interrupt sources of the INS8250 to separately activate the chip Interrupt (INTRPT) output signal. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the interrupt enable register. Similarly, by setting the appropriate bits of this register to a logic 1, selected interrupts can be enabled. Disabling the interrupt system inhibits the interrupt identification register and the active (high) INTRPT output from the chip. All other system functions operate in their normal manner, including the setting of the line status and MODEM status registers. The contents of the interrupt enable register are indicated in Table 1 and are described below.

Bit 0: This bit enables the Received Data Available Interrupt when set to logic 1. Bit 0 is reset to logic 0 upon completion of the associated interrupt service routine.

Bit 1: This bit enables the Transmitter Holding Register Empty Interrupt when set to logic 1. Bit 1 is reset to logic 0 immediately upon reading the Interrupt Identification Register.

Bit 2: This bit enables the Receiver Line Status Interrupt when set to logic 1. Bit 2 is reset to logic 0 upon completion of the associated interrupt service routine.

Bit 3: This bit enables the MODEM Status Interrupt when set to logic 1. Bit 3 is reset to logic 0 upon completion of the associated interrupt service routine.

Bits 4 through 7: These four bits are always logic 0.

MODEM CONTROL REGISTER

This 8-bit register controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM). The contents of a MODEM control register are indicated in Table 2 and are described below.

Bit 0: This bit controls the Data Terminal Ready ($\overline{\text{DTR}}$) output. When bit 0 is set to a logic 1, the $\overline{\text{DTR}}$ output is forced to a logic 0. When bit 0 is reset to a logic 0, the $\overline{\text{DTR}}$ output is forced to a logic 1.

NOTE: The $\overline{\text{DTR}}$ output of the INS8250 may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the succeeding MODEM or data set.

Bit 1: This bit controls the Request to Send ($\overline{\text{RTS}}$) output. Bit 1 affects the $\overline{\text{RTS}}$ output in a manner identical to that described above for bit 0.

Bit 2: This bit controls the Output 1 ($\overline{\text{OUT 1}}$) signal, which is an auxiliary user-designated output. Bit 2 affects the $\overline{\text{OUT 1}}$ output in a manner identical to that described above for bit 0.

Bit 3: This bit controls the Output 2 ($\overline{\text{OUT 2}}$) signal, which is an auxiliary user-designated output. Bit 3 affects the $\overline{\text{OUT 2}}$ output in a manner identical to that described above for bit 0.

Bit 4: This bit provides a loopback feature for diagnostic testing of the INS8250. When bit 4 is set to logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 1) state; the receiver Serial Input (SIN) is disconnected; the output of the transmitter shift register is "looped back" into the receiver shift register input; the four MODEM control inputs ($\overline{CTS}$, $\overline{DSR}$, $\overline{RLSD}$, and $\overline{RI}$) are disconnected; and the four MODEM control outputs ($\overline{DTR}$, $\overline{RTS}$, $\overline{OUT\ 1}$, and $\overline{OUT\ 2}$) are internally connected to the four MODEM Control inputs. In the diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit- and receive-data paths of the INS8250.

In the diagnostic mode, the receiver and transmitter interrupts are fully operational. The MODEM control Interrupts are also operational but the interrupts sources are now the lower four bits of the MODEM control register instead of the four MODEM control inputs. The interrupts are still controlled by the interrupt enable register.

The INS8250 interrupt system can be tested by writing into the lower six bits of the line status register and the lower four bits of the MODEM status register. Setting any of these bits to a logic 1 generates the appropriate interrupt (if enabled). The resetting of these interrupts is the same as in normal INS8250 operation. To return to this operation, the registers must be reprogrammed for normal operation and then bit 4 must be reset to logic 0.

Bits 5 through 7: These bits are permanently set to logic 0.

MODEM STATUS REGISTER

This 8-bit register provides the current state of the control lines from the MODEM (or peripheral device) to the CPU. In addition to this current-state information, four bits of the MODEM status register provide change information. These bits are set to a logic 1 whenever a control input from the MODEM changes state. They are reset to logic 0 whenever the CPU reads the MODEM status register.

The contents of the MODEM status register are indicated in Table 2 and are described below.

Bit 0: This bit is the Delta Clear to Send (DCTS) indicator. Bit 0 indicates that the $\overline{CTS}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 1: This bit is the Delta Data Set Ready (DDSR) indicator. Bit 1 indicates that the $\overline{DSR}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 2: This bit is the Trailing Edge of Ring Indicator (TERI) detector, Bit 2 indicates that the $\overline{RI}$ input to the chip has changed from an On (logic 1) to an Off (logic 0) condition.

Bit 3: This bit is the Delta Received Line Signal Detector (DRLSD) indicator. Bit 3 indicates that the $\overline{RLSD}$ input to the chip has changed state.

NOTE: Whenever bit 0, 1, 2, or 3 is set to logic 1, a MODEM Status interrupt is generated.

Bit 4: This bit is the complement of the Clear to Send ($\overline{CTS}$) input.

Bit 5: This bit is the complement of the Data Set Ready ($\overline{DSR}$) input.

Bit 6: This bit is the complement of the Ring Indicator ($\overline{RI}$) input.

Bit 7: This bit is the complement of the Received Line Signal Detect ($\overline{RLSD}$) input.

Interrupt Identification Register			Interrupt Set and Reset Functions			
Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Flag	Interrupt Source	Interrupt Reset Control
0	0	1	—	None	None	—
1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
1	0	0	Second	Received Data Available	Receiver Data Available	Reading the Receiver Buffer Register
0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register
0	0	0	Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Received Line Signal Detect	Reading the MODEM Status Register

Table 4
Interrupt Control Functions.

CUSTOMER SERVICE

REPLACEMENT PARTS

Please provide complete information when you request replacements from either the factory or Heath Electronic Centers. Be certain to include the **HEATH** part number exactly as it appears in the parts list.

ORDERING FROM THE FACTORY

Print all of the information requested on the parts order form furnished with this product and mail it to Heath. For telephone orders (parts only) dial 616 982-3571. If you are unable to locate an order form, write us a letter or card including:

- Heath part number.
- Model number.
- Date of purchase.
- Location purchased or invoice number.
- Nature of the defect.
- Your payment or authorization for COD shipment of parts not covered by warranty.

Mail letters to: Heath Company
Benton Harbor
MI 49022
Attn: Parts Replacement

Retain original parts until you receive replacements. Parts that should be returned to the factory will be listed on your packing slip.

OBTAINING REPLACEMENTS FROM HEATH ELECTRONIC CENTERS

For your convenience, "over the counter" replacement parts are available from the Heath Electronic Centers listed in your catalog. Be sure to bring in the original part and purchase invoice when you request a warranty replacement from a Heath Electronic Center.

TECHNICAL CONSULTATION

Need help with your kit? — Self-Service? — Construction? — Operation? — Call or write for assistance. you'll find our Technical Consultants eager to help with just about any technical problem except "customizing" for unique applications.

The effectiveness of our consultation service depends on the information you furnish. Be sure to tell us:

- The Model number and Series number from the blue and white label.
- The date of purchase.
- An exact description of the difficulty.
- Everything you have done in attempting to correct the problem.

Also include switch positions, connections to other units, operating procedures, voltage readings, and any other information you think might be helpful.

Please do not send parts for testing, unless this is specifically requested by our Consultants.

Hints: Telephone traffic is lightest at midweek — please be sure your Manual and notes are on hand when you call.

Heathkit Electronic Center facilities are also available for telephone or "walk-in" personal assistance.

REPAIR SERVICE

Service facilities are available, if they are needed, to repair your completed kit. (Kits that have been modified, soldered with paste flux or acid core solder, cannot be accepted for repair.)

If it is convenient, personally deliver your kit to a Heathkit Electronic Center. For warranty parts replacement, supply a copy of the invoice or sales slip.

If you prefer to ship your kit to the factory, attach a letter containing the following information directly to the unit:

- Your name and address.
- Date of purchase and invoice number.
- Copies of all correspondence relevant to the service of the kit.
- A brief description of the difficulty.
- Authorization to return your kit COD for the service and shipping charges. (This will reduce the possibility of delay.)

Check the equipment to see that all screws and parts are secured. (Do not include any wooden cabinets or color television picture tubes, as these are easily damaged in shipment. Do not include the kit Manual.) Place the equipment in a strong carton with at least THREE INCHES of *resilient* packing material (shredded paper, excelsior, etc.) on all sides. Use additional packing material where there are protrusions (control sticks, large knobs, etc.). If the unit weighs over 15 lbs., place this carton in another one with 3/4" of packing material between the two.

Seal the carton with reinforced gummed tape, tie it with a strong cord, and mark it "Fragile" on at least two sides. Remember, the carrier will not accept liability for shipping damage if the unit is insufficiently packed. Ship by prepaid express, United Parcel Service, or insured Parcel Post to:

Heath Company
Service Department
Benton Harbor, Michigan 49022

HEATH
ZENITH

MP MASTER
MA22781

HEATH COMPANY • BENTON HARBOR, MICHIGA

THE WORLD'S FINEST ELECTRONIC EQUIPMENT IN KIT FORM

LITHO IN U.S.A.